Use of machine learning and computational methods to parameterise process-based models and develop data driven models predicting *Miscanthus* yield

Emil Marinov Yosifov

Thesis submitted in fulfilment of the requirements for the degree of PhD

Institute of Biological, Environmental and Rural Sciences

25th September 2017

Electronic Thesis Declaration

Author Name: Emil Marinov Yosifov

Title of work: Use of machine learning and computational methods to parameterise process-based models and develop data driven models predicting *Miscanthus* yield

Department: IBERS

Research grant (if any): BB/K012509/1

Qualification/Degree obtained: PhD

Keywords: *Miscanthus*, machine learning, random forests, crop model, k-NN, scatter search, genetic algorithms

(Optional) Information Services would like to promote Aberystwyth Research Theses. If there is something you would like us to tweet, please let us know here:

……

Please sign Section A or Section B

Section A (for candidates agreeing to open access now or following an embargo period)

Details of the Work

I hereby authorise deposit of the above item in the digital repository maintained by Aberystwyth University, and/or in any other repository authorised for use by Aberystwyth University.

This item is a product of my own research endeavours and is covered by the agreement below in which the item is referred to as "the Work". It is identical in content to that deposited in the Library, subject to point 4 below.

Non-exclusive Rights

Rights granted to the digital repository through this agreement are entirely non-exclusive. I am free to publish the Work in its present version or future versions elsewhere.

I agree that Aberystwyth University may electronically store, copy or translate the Work to any approved medium or format for the purpose of future preservation and accessibility.

Aberystwyth University is not under any obligation to reproduce or display the Work in the same formats or resolutions in which it was originally deposited.

AU Digital Repository

I understand that works deposited in the digital repository will be accessible to a wide variety of people and institutions, including automated agents and search engines via the World Wide Web.

I understand that once the Work is deposited, the item and its metadata may be incorporated into public access catalogues or services, and national databases of electronic theses and dissertations such as the British Library's EThOS.

I declare/agree:

1. That I am the author or have the authority of the author/s to make this agreement and do hereby give Aberystwyth University the right to make available the Work in the way described above.

2. That the electronic copy of the Work deposited in the digital repository and covered by this agreement, is identical in content to the paper copy of the Work deposited in the Library of Aberystwyth University and the National Library of Wales, subject to point 4 below.

Electronic Thesis Declaration

3. That I have exercised reasonable care to ensure that the Work is original and, to the best of my knowledge, does not breach any laws including those relating to defamation, libel and copyright.

4. That, in instances where the intellectual property of other authors or copyright-holders is included in the work, and as appropriate, I have either:

   - gained explicit permission for the inclusion of that material in the electronic form of the Work as accessed through the open access digital repository OR
   - limited it to amounts allowed for by current legislation OR
   - established that the material is out of copyright OR
   - removed that material from the electronic version to be deposited OR
   - highlighted that material which needs to be removed from the electronic version and informed Information Services

5. That Aberystwyth University does not hold any obligation to take legal action on behalf of the Depositor, or other rights holders, in the event of a breach of intellectual property rights, or any other right, in the material deposited.

6. That Aberystwyth University reserves the right to impose an indefinite embargo should it see fit.

7. That if, as a result of my having knowingly or recklessly given a false statement at points 1, 2, 3 or 4 above, the University suffers loss, I will make good that loss and indemnify Aberystwyth University for all actions, suits, proceedings, claims, demands and costs occasioned in consequence of my false statement.

EITHER *(tick as appropriate)*

☑ I agree to my thesis being made available immediately

OR

I wish to impose an automatic embargo on public access of 2 years (this does not include bibliographic data and abstracts). Tick appropriate reason:

☐ Seeking publication

☐ Commercial sensitivity/interests

☐ Other (please specify)……

Signature ……Date: 29/10/2018

Authorisation of embargo (to be signed by Associate Dean for Research or designated nominee)

An embargo on public access of ……..years has been agreed for this work

Signed…….Date …….

Print Name…….....

Abstract

Global climate change is one of the most significant challenges faced by humanity in the 21st century. The adoption of the Paris agreement was the latest action by the international community aimed to limit its impact, by employing a range of strategies for greenhouse gas emissions mitigation. Biofuels have a high potential to reduce net CO_2 emissions, which are the primary contributor to climate change.

Miscanthus is a rhizomatous C4 grass, which has been identified as an ideal candidate for a biofuel crop. The development of simulation models of *Miscanthus* plays a crucial role in improving the crop and making it a viable product. Crop models contribute to our understanding of the biology of the plant and its interaction with the environment, and assist in decision-making in a wide range of processes, including breeding, agriculture, and environmental planning.

Crop models are unified in their reliance upon phenotypic data for parameterisation. However, the collection of phenotypic data presents a bottleneck to developing models, as it is a costly and time-consuming process. This research aims to use machine learning and other computational methods to build models and reduce the cost of data collection, without sacrificing modelling accuracy. A range of machine learning models of *Miscanthus* were developed and their precision was validated against an established crop model and real data. An optimisation method was applied to the previously developed models, which assessed the importance of commonly measured phenotypic variables and identified crucial periods of the growing season for data collection.

This research has shown that machine learning could be a powerful tool in crop modelling, for building accurate predictive models quickly and easily. It has demonstrated a methodology for minimising data collection costs, which informs experiment design. Thus, it has the potential to play an important role in *Miscanthus* breeding and agriculture.

Acknowledgements

I would like to express my special appreciation and thanks to my supervisors Lin Huang, Chris Davey and Ross King. Without their help, guidance and advice, this project would have not been possible. I would like to thank several members of the IBERS *Miscanthus* breeding team, for the assistance they gave me. Specifically, I wish to thank John Clifton-Brown for his advice and help, and Chris Ashman, Robin Warren, Chris Glover, and Marc Loosley, for their practical help and collaboration that made possible the collection of extensive phenotypic data. I would also like to thank Michal Mos and the Terravesta ltd. team for helping me collect data in their field sites near Lincoln.

I would especially like to thank Michael Squance, for the great help and guidance he provided on the numerous challenges in machine learning, statistics and biology I encountered during this project. Thanks to my fellow PhD students Odin Morón-García, Pilar Martínez-Martín, Marta Malinowska, Evangelia Stavridou, and Laura Cammarisano for their support, important advice, but especially for making the last for years enjoyable and memorable. Special thanks to Zeus, Laura Cammarisano's lovely cat, for his emotional support, despite our conversations being mostly one-sided.

I wish to give the greatest thanks to my partner Denitsa Bankova for much love, understanding and patience, my brother Alexander Yosifov for his encouragement and motivation, and my lovely parents Marin Yosifov and Krasimira Yosifova, for their selfless love and for being an endless source of inspiration.

Abbreviations

ABR 15 – a trial site of 108 F1 plant hybrids from a cross between *M. sacchariflorus* and *M. sinensis.*

ABR 46 – a trial part of the GIANT-LINK project, which included five *Miscanthus* genotypes (GNT 1, GNT 2, GNT 3, GNT 4 and GNT 5) produced through breeding in the project.

ABR 60 – trial of three planting densities (50cm, 71cm and 100cm), tested on two *Miscanthus* genotypes – Goliath and Gig-311

ABR 61 – trial established by the BSBEC BioMASS Programme. It was a randomised block design, with four *Miscanthus* genotypes planted in plots, replicated four times. All analysis in the thesis was done using the data from this trial.

APBPM – automatically parameterised BSBEC process model

BPM – BSBEC process model

BSBEC - BBSRC Sustainable Bioenergy Centre

CV – cross-validation

GA – genetic algorithm

GAI – green area index

GBM – generalised boosted model

HCK 1 – a trial part of the GIANT-LINK project, which included five genotypes (GNT 1, GNT 2, GNT 3, GNT 4 and GNT 5) produced through breeding in the project.

KNN – k-nearest neighbour

LAI – leaf area index

LER – leaf expansion rate

LM – linear model

NDVI - normalised difference vegetation index

PAR – photosynthetically active radiation

PI – proportion of light intercepted by the plant

RF- random forests

RMSE – root mean square error

RSE – root square error

RUE – radiation use efficiency

SVM – support vector machine

TPAR – total PAR on the day

Glossary of machine learning and computational methods terminology

Artificial neural network – a machine learning model based on the concept of biological neural network.

Bagging – a machine learning ensemble method characterised by resampling of training data (boostrapping), creating multiple different versions of the dataset. This is followed by training a model on each version. The result is a set of models (bag of models) used to make the final predictions, instead of just a single one.

Boosting – another machine learning ensemble method, which fits multiple models on weighed versions of the training data. Each data point is weighed depending on the prediction error of a model over it.

Bootstrapping – resampling a dataset, to create multiple different versions of it

Ensemble methods – machine learning approaches that create multiple models that provide predictions. For each data sample, predictions from all models are combined into a single final prediction.

Generalised boosted model – a machine learning model which uses boosting to create multiple models

Greedy strategy – an algorithm heuristic which makes the optimal decision at each iteration. Greedy strategies can find an optimum very quickly, but this optimum is not guaranteed to be the global optimum.

K-nearest neighbour – a machine learning model which uses its data as a model directly. Its predictions are combinations of the dependent variables of the samples nearest to the one in question.

Learner – an algorithm which builds a machine learning model

Linear regression – a machine learning model which models the relationship between dependent and independent variables in a linear way

Random forest – an ensemble bagging machine learning model which uses decision tree as its base learner.

Support vector machine – a machine learning model which uses a hyperplane to model the data.

Table of contents

List of figures

List of tables

Chapter 1: Introduction

1.1 Greenhouse gasses and the Energy Crisis

Since the middle of the twentieth century, strong evidence of significant change in global climate patterns has been collected. For that period, a rise in the temperature of the atmosphere and oceans, a reduction in the amounts of snow and ice, and the rise of the sea level have been observed. The 2014 report on climate change prepared by the Intergovernmental Panel on Climate Change (IPCC) (IPCC, 2014) lists the increased atmospheric concentrations of greenhouse gases (GHG), particularly carbon dioxide, methane, and nitrous oxide, as being extremely likely (95-100% likelihood) to be the dominant cause for global warming. Since the industrial revolution, the anthropogenic GHG emissions have been increasing, reaching 49 $GtCO_2$-eqivalent/year in 2010. The most influential anthropogenic activities are fossil fuel combustion and industrial processes that release CO_2 contributing to 78% of the emission increase between 1970 and 2010. In 2015 the emissions due to these factors have finally shown zero growth (Le Quéré *et al.*, 2016). However, fossil fuels remain the main sources of energy, accounting for 78.3% of the global energy production (REN21, 2016).

There is a growing need for alternative fuel sources, that do not exacerbate global warming. Concerns over climate change has led to an increase in the renewable energy consumption worldwide. Currently the global energy market share of renewables is 19.2% (REN21, 2016). As part of "Europe 2020", EU's ten-year growth strategy, the EU has set a target of 20% of the total energy to come from renewables (European Parliament, 2009). The Paris agreement signals the commitment of the signatories to reducing GHG emissions and to keeping the global temperature below 2°C above pre-industrial levels (United Nations/Framework Convention on Climate Change, 2015). To meet these requirements, the renewable energy industry will require innovations, improvement and expansion.

1.2 Renewable energy

Renewable energy comes from multiple sources including biomass, hydropower, geothermal, wind and solar. Renewable energy sources (RES) are considered unlimited because they are naturally replenished quickly enough to meet energy requirements. They have a potential for generating massive amounts of energy, which greatly surpass anything that fossil fuels can supply. However, there are numerous difficulties which need to be overcome, before renewable sources can be used effectively (Dincer, 2000). Addressing the problems of improving the efficiency of energy collection and conversion, lowering the cost, and increasing the reliability of renewable energy sources has led to considerable progress in making renewable energy a viable alternative.

1.2.1 Traditional biomass

Biomass is the first source of energy known to humans, and has been used since ancient times (Ferreira, Moreira and Monteiro, 2009). Its major advantage is that it can be stored, whereas most other renewable energy sources are intermittent. Today, traditional biomass makes up 46% of the renewable energy consumption. It usually encompasses burning fuelwood, dung and crop residues and is mainly used for cooking and heating in the developing countries (Sims *et al.*, 2006; Gurung and Oh, 2013; REN21, 2016).

While combustion of biomass releases greenhouse gasses, like fossil fuels, it carries different net carbon emissions. A plant fixes carbon from the atmosphere, which becomes part of its biomass. By burning the biomass, this carbon is released and would later be fixed by another plant that can also be used as a fuel source. This cycle continues indefinitely, and theoretically, leads to a system that has zero net carbon emissions, as no new carbon is added. Fossil fuels, on the other hand, disturb this balance by releasing new carbon atoms that have been stored in the ground for millennia. However, biomass burning can cause environmental problems if used unsustainably. Failure to replant the harvested biomass can lead to desertification of the land and depletion of the biomass resource (Akinbami, Salami and Siyanbola, 2003). The use of dung cake and crop residues as fuel and not as fertiliser could lead

to the reduction of organic matter present in the soil and to its erosion (Reijnders, 2006).

Along with environmental problems, the inefficient burning of traditional biomass causes health issues as well. Its widespread domestic use in developing countries has raised concern over its smoke emissions, which contain several known carcinogens as well as gaseous pollutants. The health risk has been estimated to be equivalent to the risk from smoking tobacco (de Koning, Smith and Last, 1985). A more recent study reported on open fire cooking causing issues with respiratory health, eye discomfort, headache, and back pain (Romieu *et al.*, 2009).

Finally, traditional biomass is produced and used in an unsustainable manner. In developing countries biomass is often used without replanting which leads to deforestation (Goldemberg and Teixeira Coelho, 2004). The methods of utilising the energy are also very inefficient. Using open fires, traditional cooking stoves, furnaces or kilns consumes large amounts of biomass energy and is wasteful. Replacing them with their modern and efficient equivalents has the potential to dramatically reduce wasted energy (Bhattacharya *et al.*, 1999).

1.2.2 Modern biomass

Modern biomass is defined as the sustainable production and utilisation of biomass (Goldemberg and Teixeira Coelho, 2004). It widens the range of sources and improves on the efficiency of conversion to energy. Modern biomass can be divided into three groups depending on the origin and conversion process: first, second and third generation.

First generation feedstock encompasses all food crops that can be converted into energy like corn, wheat, sugarcane and oilseeds. The main products are biodiesel, ethanol and biogas (Naik *et al.*, 2010). Because of its abundance, first generation biomass is the most established modern biofuel source, but it competes for arable land with the food market and can cause an increase in food prices, making it unsustainable.

The second generation consists of lignocellulosic biomass produced from energy crops or organic leftovers from agricultural and forestry processes (Srirangan *et al.*,

2012). Energy crops are non-edible and are cultivated specifically for energy production. They can be grown under poor environmental, climate and weather conditions and have a potential to meet a considerable proportion of the energy demand without competing with food crops for land (Fischer, Prieler and van Velthuizen, 2005; Cai, Zhang and Wang, 2011; Valentine *et al.*, 2011). Candidate species are selected based on their desirable characteristics: fast growing rate, high yield potential, stress tolerance, etc. The biomass can be burned directly or converted to biofuel through several thermochemical or biochemical processes. A number of challenges have to be overcome before an energy crop can become a viable large scale product, like improving the crop through breeding, developing suitable agricultural and harvesting techniques, optimising the biomass product for transport and attracting potential growers (Mitchell *et al.*, 2016; Clifton-Brown *et al.*, 2017).

Third generation biomass makes use of algae due to its exceptional ability to consume large amounts of CO_2 and produce substantial quantities of biomass (Singh, Olsen and Nigam, 2011; Maity *et al.*, 2014). Algae can be grown in a bioreactor, removing any need for agricultural inputs or large areas of land (Srirangan *et al.*, 2012). They can be converted into various biofuels using processes like liquefaction, pyrolysis, anaerobic digestion, etc.

1.2.3 Other renewable energy sources

There are several other renewable sources of energy that humanity has learned to utilise to different extents. Geothermal, solar, wind and hydro energy are abundant, widely available, and have the potential to meet global energy demand many times over (Jacobson, 2009).

Geothermal energy

There is practically inexhaustible amounts of thermal energy stored within the Earth's interior (Barbier, 2002). However, most of it is inaccessible due to it being too deep under the surface. Some can be utilised only in certain places, where heat accumulates close enough to the surface to allow extraction of hot water or steam though drilling (Jacobson, 2009). This energy source is common in places near volcanic or fault activity. It could be used directly for heating residential buildings or

generating electricity using a geothermal power plant (Hammons, 2003). In the latter case, direct and indirect GHG can be released, unless a binary system is used, where water from the ground is kept in an enclosed pipe, not allowing any gases to escape into the atmosphere. The water heats an organic fluid, which in turn drives a turbine generating electricity. Thus, with the right methodology geothermal energy is an environmentally friendly method of power generation (Ellabban, Abu-Rub and Blaabjerg, 2014). Geothermal technology is mature, having the experience of over 100 years of commercial exploitation (Fridleifsson, 2003).

Hydropower

Hydropower is any power generated using moving water, mainly from rivers, although other sources like waves and tides have also been used (Ellabban, Abu-Rub and Blaabjerg, 2014). In 2014 it accounted for 3.9% of global energy consumption (REN21, 2016). It is a mature and proven technology that has been used since the 19th century (Frey and Linke, 2002). The simplest method of utilising hydropower is Run-of-River (RoR) hydropower plants where the flow of the river is used to turn a turbine and generate electricity. The output depends on the available flow of the river and could be affected adversely by lack of precipitation or river runoff. Reservoir-type hydropower plants use dams to collect water in a reservoir by controlling the flow of water, releasing more when more energy is required. This reduces their dependency on river flow availability and widens their possible uses. Water flow could be reduced, giving a steady supply of energy, or water could be stored and released, generating electricity for peak loads (Egré and Milewski, 2002). A third type of plant called pumped storage hydropower plants can be used to store energy by pumping water from a lower to a small, usually man-made upper reservoir during low energy demand and allowing it to flow back, generating electricity, during peak times (Ellabban, Abu-Rub and Blaabjerg, 2014). While some hydropower plants like small RoR have a minimal environmental impact, reservoir-type plants bring ecological changes to the river as they convert it into a lake-type environment. The potentially significant environmental impact is a major problem preventing the building of more pumped-storage plants, which could be addressed through site-specific design solutions (Leonhard and Grobe, 2004; Spahić *et al.*, 2007).

Solar energy

Solar energy generation encompasses methods for using sunlight to provide either thermal or electrical energy. Solar heating and cooling systems use solar energy to heat a fluid which is then used for heating purposes, to power chiller devices for cooling or to drive natural ventilation devices (Chan, Riffat and Zhu, 2010; Khanal and Lei, 2011). Solar energy could be directly converted to electricity through the use of photovoltaic (PV) systems (Ellabban, Abu-Rub and Blaabjerg, 2014). They are based on the PV cell, which is a device that uses the photovoltaic effect to produce direct current when exposed to sunlight. Multiple PV cells are combined into a PV module, which outputs between 50 and 200W. PV systems are very scalable as multiple PV modules can be connected to further increase the output power. Another method of producing electricity using solar energy is by concentrating solar power (CSP). CSP usually uses a system of mirrors to concentrate light, which heats a liquid, solid or gas, which is then used to generate electricity. One advantage of PV over CSP is that PV systems do not need direct sunlight to work, they still generate electricity from diffuse light in cloudy weather. However, the main disadvantage of both PV and CSP systems is that they can only collect energy during the daylight. In recent years the use of PV has been growing rapidly, with global capacity reaching 227 GW in 2015 (REN21, 2016). In comparison, the market share of CSP is much smaller, with global capacity of 4.8 GW, and the rate of growth in 2015 has slowed.

Wind energy

Wind energy has been used for centuries, ever since windmills were first invented (Kaldellis and Zafirakis, 2011). They were improved in the 19th century when the first electricity producing turbines were built. Recently, concerns over climate have renewed the interest in wind energy, and in recent years its market share has been growing rapidly, with global capacity reaching 433 GW (REN21, 2016). Wind energy is converted by using the kinetic energy of moving air to drive an electric generator, producing electricity (Ellabban, Abu-Rub and Blaabjerg, 2014). Recent improvement efforts have focused on the efficiency of capturing wind energy, as current wind turbines utilise only 40-50% of what is available. Other aims include reducing

manufacturing costs and making turbines more cost-effective by using lighter materials, increasing the size of the turbine and the reliability of the system.

Wind energy is a clean and cheap source of energy, but is strongly dependent on local meteorological conditions. Wind turbines require constant strong wind, which is not always available, and the roads they are transported on before assembly impose a restriction on their size. Finally, wind farms may occupy large amounts of land. Offshore wind farms have been developed to tackle these drawbacks, by placing the wind turbines in the sea, where stronger winds are available, land use is not an issue, and as they are transported by sea, there is no limit on their size (Breton and Moe, 2009). The current global capacity (2018) of offshore wind farms is 12 GW (REN21, 2016), which is only a small portion of all potential wind energy.

1.2.4 Moving towards renewable energy

Due to the individual advantages and disadvantages of each renewable energy source, the most feasible option for becoming independent from fossil fuel is to make use of all available renewable energy, and develop and improve each methodology. It has been demonstrated that starting with multiple small-scale options for reducing GHG emissions and gradually increasing them over time can prevent fossil fuel emissions from increasing and eventually completely offset them (Pacala and Socolow, 2004).

Some renewable energy sources like wind and solar provide a fluctuating power supply, necessitating the use of another backup source for load balancing (Steinke, Wolfrum and Hoffmann, 2013). It has been suggested that such intermittent sources could be combined to meet power requirements, by decentralising power generation and connecting all power sources together over a long distance in a power distribution network called SuperGrid (Battaglini *et al.*, 2009). This idea relies on the principle that generated power from different locations will peak at different times, cancelling out the interruptions and providing a steady average supply.

One example of a supergrid is the North Sea Countries' Offshore Grid – a supergrid in development that interconnects offshore wind systems and power systems in Northern Europe. In 2010, an agreement between ten European countries started its

development. By 2017, the total offshore wind capacity for Europe was 15.8 GW and was forecasted to increase to 25 GW by 2020 (Gorenstein Dedecca and Hakvoort, 2016; Wind Europe, 2017).

Building a supergrid is a substantial undertaking which is likely to take a long time as additional installations will need to be built to collect energy over a wide area, and the present electric grid would have to be expanded considerably to transmit the generated power over long distances (Delucchi and Jacobson, 2011; Jacobson and Delucchi, 2011). Thus, biomass is very likely to play an important role in the future as a carbon-neutral backup source, offsetting fossil fuel emissions.

1.3 *Miscanthus* as an energy crop

Research efforts into novel energy crops have sparked interest into using perennial rhizomatous grasses for that purpose, because of the high calorific value of their biomass, as well as the possibility to improve biomass quality through a late harvest. Extensive research has been done on the four most promising candidates: *Miscanthus,* reed canary grass (*Phalaris arundinacea*), giant reed (*Arundo donax*) and switchgrass (*Panicum virgatum)* (Lewandowski *et al.*, 2003). Grasses with the C4 photosynthetic pathway, like *Miscanthus* and switchgrass are generally considered to have an advantage over C3 grasses (reed canary grass and giant reed) because they use nitrogen more efficiently and have higher yield for CO_2 uptake (Schmitt and Edwards, 1981; Ehleringer and Pearcy, 1983). The disadvantage of most C4 species is that their performance drops under low temperatures and they have not evolved to live under cold climates. A notable exception to that rule is the *Miscanthus* genus (Naidu *et al.*, 2003; Sage and Kubien, 2007).

Miscanthus originates from eastern Asia and was first commercially used in Europe by the Danish botanist Aksel Olsen in 1935 (Linde-Laursen, 1993). The species he brought from Japan was observed to have vigorous growth and was named *M. x giganteus.* At first it attracted attention as a source of fibre and as thatch material, but later its potential as a source of bioenergy was recognised (Jones and Walsh, 2013; Clifton-Brown, Schwarz and Hastings, 2015). There have been extensive studies of *M. x giganteus* in many environments across Europe, on multiple aspects of the

crop, from yield potential to practices for propagation, management, harvesting and breeding (Lewandowski *et al.*, 2003). Some of its advantages that make it good for biomass production are its low fertiliser and pesticide need, as well as high energy use efficiency (yield energy per energy required to grow the plant) (Lewandowski and Schmidt, 2006). It has a relatively high content of lignin which is advantageous for two reasons (Ververis *et al.*, 2004). The high quantity of carbon in lignin means the plant material has a high heating value. Also, lignin allows the stems to stand upright under low water conditions, which makes it possible to employ a late harvest strategy to allow the biomass to dry before it has been harvested (Lewandowski *et al.*, 2003). Biomass quality is determined by the content of chloride (Cl), potassium (K), nitrogen (N), sulphur(S) and ash, all of which can contribute to emissions of harmful gasses, and corrosion or fouling of the combustion chamber in a power plant. They are all present in low amounts in *Miscanthus* biomass, but Cl, K and ash are further reduced by late harvest (Lewandowski and Kicherer, 1997; Iqbal *et al.*, 2017).

The success of *Miscanthus* as an energy crop is hindered by the lack of cheap and effective propagation methods (Xue, Kalinina and Lewandowski, 2015). Currently, the most established approach is rhizome propagation. Rhizomes of existing plants are split into small pieces and planted back into the soil. This could be done using standard agricultural equipment, but rhizome quality is better if rhizomes are split by hand. This approach is labour-intensive and would benefit from developing specialised equipment to mechanise the process. Rhizome propagation is limited by the amount of rhizome each parent can produce, which could create issues with large scale planting. Rhizome is also used in the rhizome-derived plant propagation method, where plantlets from rhizome pieces are grown in a glasshouse and transplanted into the field. It is more expensive, but its multiplication ratio (number of parents: number of produced plants) is 1:30, an improvement over the ratio for direct rhizome propagation, which is 1:10. The difference in cost between the two method comes from the additional labour and energy required to produce rhizome-derived plants.

Miscanthus can also be propagated from parts of the stem, which is the standard method of propagation in sugarcane (*Saccharum officinarum* L.) (Boersma and

Heaton, 2012), a closely related species of perennial grass. Stems are cut in to small pieces, while ensuring that there is an axillary node in the middle of each piece. They are then placed in pots and covered completely in potting media. Subsequently new roots and shoots grow out of the piece, and the resulting plantlet can function independently from the parent plant. As with rhizome-derived plants, plantlets are grown under favourable conditions in greenhouses and transplanted into the field. The multiplication ratio for stem propagation is 1:120 (Xue, Kalinina and Lewandowski, 2015), but it is as labour-intensive as and only slightly cheaper than rhizome-derived propagation. In vitro techniques, where axillary buds are placed within media that induces shoot and root production, have been applied for *Miscanthus* propagation (Atkinson, 2009). Their multiplication ratio is significantly higher at 1:960 but the method is prohibitively expensive.

Seed propagation is the most inexpensive out of all possible methods. Its low cost and extremely high multiplication ratio (1:1172) make it the most attractive propagation approach for large-scale industrial planting (Xue, Kalinina and Lewandowski, 2015; Clifton-Brown *et al.*, 2016; Hastings *et al.*, 2017). Currently the method is not used commercially because it cannot be used for *Miscanthus x giganteus*, as it is a triploid hybrid, and so does not produce any seed. However, it can be used for any new *Miscanthus* varieties that produce seed, which are being developed and improved through selective breeding. The seed set rate, which is the average number of seeds divided by the average number of florets, is important for the success of seed propagation and some research has been done on differences in the rate between species as well as conditions contributing to its increase (Adati, 1958; Deuter, 2000). Seed establishment is another aspect that requires improvement. This can be achieved by growing plantlets under controlled conditions and transplanting them. The seeds can be stored for a maximum of two years, which surpasses the storage capacity of any other method. Seed propagation also has the added advantage of not damaging the parent plants.

1.4 Miscanthus breeding

Plant breeding is the practice of selecting plants with favourable characteristics, which when crossed, produce new varieties that are more suitable for their intended

purpose. It was best defined by (Bernardo, 2002) as "the science, art and business of improving plants for human benefit". The success of *Miscanthus* as a bioenergy crop depends both on the development of improved and specifically tailored agricultural techniques, and the breeding of new enhanced varieties that would realise its potential for being a high yielding, low-maintenance and economically viable energy crop. It has been demonstrated that *M. x giganteus* is not only costly and impractical to propagate on large scale, but also not consistently the highest performing genotype (Clifton-Brown *et al.*, 2001). While it performs well in England and Germany, its yield has been surpassed by other genotypes in locations further north and south (Kalinina *et al.*, 2017). *Miscanthus* is distributed over a wide geographic area in Asia and exists under a range of climatic conditions, which can be seen in the geographical patterns in the genetic and phenotypic variability between genotypes (Slavov *et al.*, 2013; Li *et al.*, 2016). These differences suggest that no single genotype exists that performs best at all locations, but instead different sets of traits should be combined through breeding a variety of genotypes, to produce cultivars specifically adapted to each climate.

1.4.1 Advantageous traits

Due to the need for *Miscanthus* hybrids specifically bred for a given environment, recent research has focused on *Miscanthus* genotypes' response to stresses and tolerance of environmental conditions like drought (Clifton-Brown and Lewandowski, 2000; Malinowska, Donnison and Robson, 2016), soil salinity (Stavridou *et al.*, 2017), low temperatures (Naidu *et al.*, 2003; Peixoto, Friesen and Sage, 2015), frost (Friesen *et al.*, 2014), oxidative stress and to the presence of aluminium and heavy metals in the soil (Ezaki *et al.*, 2008). Identifying stress tolerant genotypes and understanding the mechanism behind stress responses provides vital information for breeding efforts.

Beyond plant survivability, breeding can be used to increase the quantity of biomass produced. Yield is a complex trait, which is a result of the interaction between multiple phenotypic traits and the environment over the whole growing season. Several studies have been conducted on traits contributing yield, which have identified plant height, height of the tallest stem, stem diameter (Clifton-Brown *et*

al., 2001; Robson, Jensen, *et al.*, 2013; Kalinina *et al.*, 2017), and canopy duration (number of days between plant canopy establishment and senescence) (Robson, Farrar, *et al.*, 2013) to be highly correlated with yield. Another study has suggested that stem count and plant diameter are the most indicative traits for yield over the first three years of development (Jeżowski, 2008). Identifying genotypes with advantageous traits and crossing them together is a strategy employed by breeders to produce new varieties or to improve existing ones.

Biomass quality is the third aspect of *Miscanthus* that requires development, to make it a competitive and successful energy crop. While late harvest brings the most significant improvement in quality, breeding could also make a significant contribution. For example, *M. sinensis* has been found to contain 3-8 times lower amounts of K and Cl compared to *M. x giganteus*, although there is a lot more variation between *M. sinensis* genotypes (Jørgensen, 1997). Similar results were presented in another study, which also lists early flowering time and senescence rate as crucial traits for biomass quality for combustion (Clifton-Brown and Lewandowski, 2002). The authors theorised that the higher biomass quality in *M. sinensis* may have been due to its early senescence, which would have triggered movement of nutrients from the stems to the rhizome before stems are killed by frost. Unfortunately, *M. sinensis* is not the best performer in terms of yield because its stems are thin. A later study in *Miscanthus* biomass combustion quality observed a positive correlation between Cl and K concentrations and stem thickness (Iqbal and Lewandowski, 2014), which suggests that biomass quality and quantity are conflicting traits, where improving one impacts negatively on the other. The candidate genotype for breeding suggested by this study is a *M. sacchariflorus* which strikes the best balance between having favourable biomass qualities and high yield.

The chemical composition needed for high quality biomass changes depending on the energy conversion route used, which becomes clear from a study into genotypic variation in cell wall composition between *Miscanthus* genotypes (Hodgson *et al.*, 2011). It argued that the high holocellulose (cellulose and hemicellulose) and low lignin content exhibited by one of the investigated *M. sinensis* genotypes made it more appropriate to be converted to bio-oil or alcohols, whereas a *M. sacchariflorus*

genotype that had low holocellulose and high lignin was more fitting for combustion. This shows that the intended use of the biomass needs to be considered when breeding new varieties, as it determines which traits would be considered advantageous for biomass quality.

1.4.2 Breeding methods

Breeding history

Plant breeding has a long history starting with the emergence of agriculture approximately 11,000 years ago, when crop plants were first domesticated (Xu, 2010). Plant domestication involves the artificial selection of plants by humans, to retain favourable characteristics that make them better adapted for growing and their end purpose: as food, or source of fuel, fibres, medicines, building material, etc. The continuous selection of plants has altered them genetically, because it gives priority to rare mutant alleles that contribute to desirable traits, which are not necessary in the wild, spreading them throughout the population of domesticated plants. Domestication can be thought of as guided evolution where natural selection is diminished by growing the plants in cultivated fields, and artificial selection is applied by removing the plants that do not exhibit the required characteristics. The classical approach to make artificial selection is to use phenotypic data directly, a process known as phenotypic selection (PS). This method relies heavily on the collection of accurate phenotypic data, to assess the value of each plant in a population and make selections. Although newer selection methods are available, which are discussed later in this chapter, phenotypic selection still plays an important role in breeding programmes (Koebner and Summers, 2003).

Along with selection, a crucial step for plant breeding is reproduction. The discovery of sex in plants is credited to Camerarius (1694) and the first systematic study of hybridization was done by Kölreuter (1761) (Zirkle, 1934). Hybridization is the cross-fertilization between individuals from two populations that can be distinguished by heritable characteristics (Rieseberg and Carney, 1998). It allows breeders to select individuals from different populations, expanding the possible sources of desirable traits. In 1865 Gregor Mendel established the basic rules of heredity through his pea plant experiments, although his findings were not applied to plant breeding until the

early 20th century (Xu, 2010). At that time, important breeding techniques like pedigree breeding, backcross breeding and mutation breeding were developed. Qualitative genetics, the study of the genetics behind traits with continuous variation (height, stem count) rather than discrete, or qualitative traits, emerged from Fisher's work in 1918, where he applied Mendelian principles to the inheritance of quantitative traits (Walsh, 2001). Quantitative genetics have further evolved more recently with the development of molecular markers which allow for expressing a complex quantitative trait as a collection of Mendelian factors.

Genetic markers

Molecular markers are specific locations on a chromosome that reveal polymorphisms at either the protein (biochemical markers) or DNA level (DNA markers) (Kumar, 1999). Biochemical markers are proteins that have been produced by gene expression and separated through electrophoresis. Differences in gene sequence between individuals affect protein structure, which is detected when separating the proteins. DNA markers are loci containing polymorphisms in the DNA sequence. These are caused by DNA mutations like insertion, deletions or substitutions, and there are many methods for their detection (Collard *et al.*, 2005). Examples of such methods include restriction fragment length polymorphism (RFLP) (Beckmann and Soller, 1986), random amplified polymorphic DNA (RAPD) (Williams *et al.*, 1990), simple sequence repeats (SSR) (Taramino and Tingey, 1996) and amplified fragment length polymorphism (AFLP) (Vos *et al.*, 1995). More recently these methods have fallen out of favour to sequence-based methods, which are used to detect single-nucleotide polymorphisms (SNPs). A SNP is a difference in a single nucleotide base pair between two individuals (Xu, 2010). SNPs are spread through the entire genome, and are the most abundant markers available, as evidenced by the discovery that SNPs account for 90% of the variation in the human genetic sequence (Collins, Brooks and Chakravarti, 1998).

The location of markers is crucial to their usefulness. Each marker is located at a specific position on the chromosome called a "locus". There are regions in the genome that are linked to or contain genes that contribute to quantitative traits, called quantitative trait loci (QTL). Different versions of the genes (alleles) in the QTL

result in difference in phenotype. If a marker is close enough to a QTL, polymorphism in the marker may be correlated with different alleles (Collard *et al.*, 2005). This is so because the closer to a QTL a marker is, the smaller the probability that a recombination will occur between them during chromosome crossover. A marker that is associated with a QTL or gene is said to be linked to the trait, and the act of finding these linked markers is called QTL mapping. Thus, markers do not influence phenotype directly, as they are usually located near a QTL. They serve as indicators of chromosomal regions of the genome that code for a trait of interest. They are also used for determining the genotype of the individual. This is especially useful to breeding because it provides new methods for performing selection.

QTL mapping is applied to specific mapping population of filial 2 (F2) hybrids, produced from two parents. While it is a powerful method, a major limitation to the method is that it could only detect QTL for traits that segregate between the parents of the population (Korte and Farlow, 2013). With the increase of SNP data density, containing a wider coverage of the chromosome, another method for finding marker-trait associations, genome wide association studies (GWAS), becomes feasible. It operates by calculating the association between each marker and phenotypic trait of interest and does not suffer from the limitation of QTL mapping. However, the power of GWAS to identify marker-trait association is limited by the trait variance in the observed population. GWAS also has the issue of "missing heritability". Heritability is a measure of variation in the phenotype due to genetic variance. GWAS underestimates the heritability of a phenotypic trait, which means it has not captured the full additive genetic variance which contributes to that trait (Zaitlen and Kraft, 2012).

Phenotypic data is central to both QTL and GWAS, as both methods look for genetic causes to variation in the phenotypic trait of interest. The accuracy of QTL and GWAS depends on the phenotypic data quality (Chang *et al.*, 2006). Errors in the phenotypic information could cause these two methods to fail to detect regions of the DNA linked with a trait. Because of this, phenotypic measurements collected for association studies may have to be repeated as this reduces phenotypic variance and gives a better chance for the two methods to detect genetic effects.

Marker assisted selection

Markers have several uses in a plant breeding context. Genotyping individual plants is a way of identifying and characterising varieties, through their DNA markers. This is important for breeding programmes as it helps protect breeders' intellectual property rights (Xu, 2010; Korir *et al.*, 2013). Markers can also be used to assess the genetic diversity of any new breeding material being added to the genetic base. Genetic diversity is important in a breeding programme as it is a source for new and desirable traits (Collard and Mackill, 2008).

The biggest contribution of QTL and markers towards breeding is a new methodology of breeding selection called marker-assisted selection (MAS). Individuals from a population are selected indirectly using their markers rather than phenotype. Phenotypic measurements are costly and time-consuming, and phenotypic traits are not the best selection criteria when improving complex inheritance or low penetrance traits like yield (Xu and Crouch, 2008). MAS provides an advantage for traits that are hard to detect or quantify, like resistances, stress tolerance and quantitative traits (Mohan *et al.*, 1997). For example, with phenotypic selection every new variety will need to be tested under abiotic stresses, which requires growing the individual under the right stress conditions. With markers, simply genotyping the individual would reveal stress tolerance. Thus, MAS can be used for pyramiding, i.e. combining favourable genes for multiple traits simultaneously, much more precisely, and significantly quicker. Finally, this method could be used to screen and eliminate hybrid lines that have not inherited the right combination of trait during early plant developmental stages, which is especially useful when selecting for traits expressed later in the growing season (Collard and Mackill, 2008). This allows for more focus on fewer high-performing strains. The role of phenotypic data has shifted from being used directly for selection in PS, to being used to identify genetic markers which are then used for selection in MAS. This, however, does not diminish its importance, as good quality phenotypic data is crucial to the establishment of marker-trait associations and therefore to the successful application of MAS (Ribaut and Ragot, 2007).

MAS is a promising methodology, but it does have its limitations. While it has been demonstrated to improve traits that are controlled by a few large effect alleles, it has failed when multiple small effect alleles influence the trait. In the latter case, it can be a lot harder to find QTL, small-effect QTL could go undetected and the results may be unreliable and biased (Jannink, Lorenz and Iwata, 2010). The establishment of biparental populations required for QTL mapping is costly, which limits its size and the accuracy of the results. Also a biparental population would not capture the full allelic diversity of the breeding population, making multiple mapping populations necessary (Heffner, Sorrells and Jannink, 2009).

Genomic selection

The solution to the problems that MAS faces is to move away from looking for single markers that improve the crop, and to account for the collective effect of all markers. Genomic selection (GS) achieves that by building a model on a training population of phenotyped and genotyped individuals, that scores individuals with genomic estimated breeding values (GEBVs). While GEBVs do not contain information on the functionality of the underlying genes, they reflect an estimated importance of the individual for breeding improved varieties, and is thus a great selection criteria (Heffner, Sorrells and Jannink, 2009; Jannink, Lorenz and Iwata, 2010). GEBVs for new individuals are predicted by the model only based on their genotypic data.

A typical breeding cycle follows four steps: crossing plants to get new breeding material, genotyping the new plants, calculating GEBVs using the model, and making selections based on their score. The GS model can be updated each time new phenotypic and genotypic information is obtained. The accuracy of GEBVs depends on how representative the training population is of the breeding population. While phenotypic data plays a different role in this method compared to the one in PS, it is still a very important one. As phenotyping is a time-consuming and costly process, and the cost for genotyping is decreasing, GS changes the role of phenotypic data from being a direct selection target to being training data for the GS model (Jannink, Lorenz and Iwata, 2010). The quality and quantity of phenotypic data is a defining factor for the accuracy of the resulting GS model and to the success of GS.

1.4.3 *Miscanthus* breeding programme

In 2001, a study of the biomass production of *Miscanthus* genotypes in multiple locations in Europe revealed that it would not be optimal for a single *Miscanthus* genotype to produce all biomass across the continent. This was due to genetic differences between genotypes that made some of them perform poorly in some regions but exhibit top performance in others, which had different climatic conditions (Clifton-Brown *et al.*, 2001). The future success of *Miscanthus* as a biomass crop depended on efforts into *Miscanthus* breeding, that would combine the various desirable phenotypic traits and develop new hybrids that are tailored for given environmental conditions.

In 2004 a *Miscanthus* breeding programme funded by DEFRA was established in the Institute of Biological, Environmental and Rural Sciences (IBERS) in Aberystwyth (Clifton-Brown, Schwarz and Hastings, 2015). The programme aims are to release commercial *Miscanthus* hybrids and to develop the relevant agronomy. It follows four stages: collect germplasm with useful traits from the wild, use crossing to create new hybrids, evaluate their performance in a range of environments, scale up planting and develop the necessary agronomical practices for establishment, management and harvesting. The initial germplasm was provided from various sources from Europe, including Kew Botanic gardens collections and European funded projects like EMI and BIOMIS (Clifton-Brown *et al.*, 2008). In 2007, multiple collections of germplasm in Asia have added to the programme a large genetically variable *Miscanthus* germplasm collection. In 2011, a joint project between public (BBSRC and DEFRA) and private (CERES, Blankney Estates, E.ON and NFU) partners called GIANT-LINK began, which greatly advanced the breeding programme. Large scale seed production of advantageous parental crosses, discovered through testing in Aberystwyth, was started by CERES in 2013. The new hybrids are tested in multiple locations across Europe.

1.4.4 Crop modelling

Crop models are software tools to simulate the growth and development of crops, with a wide range of applications in agriculture (Oteng-Darko *et al.*, 2013). They are a way of testing scientific hypotheses, through simulations of the complex interaction

between weather, soil and plant, which can lead to further understanding of the crop biology (Di Paola, Valentini and Santini, 2016). They aid environmental, agronomical and business decision-making by simulating the outcomes of growing a crop under certain environmental and agronomic conditions, in terms of crop performance (yield) and environmental impact.

One use of crop models in breeding programmes is the formulation and description of hypothetical plants (ideotype) that are perfect for a target environment (Donald, 1968). This can provide breeding programmes with new breeding targets and estimate the likely performance of these ideotypes through simulation. Crop models can also be employed to simulate performance of new hybrids produced by the breeding programme, in different conditions, to find suitable locations for planting and give an estimate of expected yield from each one.

Several crop models have been developed for *Miscanthus* in the last 20 years. They are based on a widely used approach in crop modelling, where yield is modelled in terms of the amounts of light, which is absorbed by the plant and converted into plant matter (Monteith and Moss, 1977). One basic *Miscanthus* model following this approach (Clifton-Brown *et al.*, 2000), has been expanded to consider environmental and developmental factors such as temperature, soil water and flowering, which resulted in the MISCANMOD crop model (Clifton-Brown, Stampfl and Jones, 2004). A newer MISCANMOD, called MISCANFOR, was later developed, which further refined the description of the crop system and took into account additional environmental factors (Hastings *et al.*, 2009). These two models have been used for estimating potential yield of the *M. x giganteus* hybrid for locations across Europe.

An important part of the development of crop models is their parameterisation, which converts data on the observed crop behaviour into a set of model parameter values. The data provided to the model for that process contains phenotypic and environmental measurements which capture the growth and development of the crop under the environmental conditions (rainfall, temperature, etc.). The parameter values describe the phenotypic behaviour of the crop and allow the model to simulate crop growth and development. Collecting accurate phenotypic data that captures the phenotypic variation in sufficiently high resolution is a major step to

parameterising the model. The quality of the phenotypic data influences the accuracy of the parameter values, and therefore the accuracy of model simulations.

The parameterisation of crop models is also a laborious process, which requires direct human intervention. This, along with the lack of extensive phenotypic data on a wide variety of genotypes, limits the number of genotypes that a crop model can be parameterised for. However, if this difficulty is overcome, a wider range of applications of crop models could be conceivable, including the use of their parameter values as target traits for selection in breeding.

1.5 Objective of the research

Phenotypic data plays a key role in crop breeding and modelling and is vital for the success of these tasks. However, the collection of phenotypic data requires a large investment of time and funding which presents a bottleneck for breeding and modelling efforts. The aim of this research is to investigate ways into optimising the collection of phenotypic data, to retain the quality of the collected data while minimising the collection cost. The main objective is to use machine learning and other computational methods to investigate ways to plan optimal data collection strategies in a *Miscanthus* crop modelling context.

Another objective is to identify ways of improving crop models by employing computational methods that allow their automatic parameterisation. This will make it easier to expand models for simulating new genotypes in the future. The final objective is to demonstrate the power of machine learning methods to develop accurate and robust crop models. Importance will be placed upon rapid model development and prediction accuracy and model flexibility.

To achieve the three objectives: data collection optimisation, automatic parameterisation of crop models and developing machine learning crop models, this thesis aims to answer the following scientific questions:

- Which phenotypic traits and environmental variables contain the most information for predicting *Miscanthus* yield?
- Is there a way to determine how much/how little data is required to successfully parameterise a *Miscanthus* crop model?
- Is there a way to automate the parameterisation of crop models, and make it easier to expand them to new genotypes?
- Do machine learning methods have a role to play for developing useful, robust and accurate crop models?

1.6 Structure of thesis

This thesis contains six chapters. Chapter 1 is the introduction, which describes the background of this research. This covers the problem of climate change, the need for renewable energy sources, and the role of *Miscanthus* as an energy crop, as well as challenges that need to be overcome in the future. Chapter 2 is an overview of the methods used in this research, as well as data collection protocols and field trial configurations for the data gathered for this thesis.

The automated parameterisation procedure created for a *Miscanthus* crop model from a previous publication is described in chapter 3 (Davey *et al.*, 2015). The work includes the validation of this procedure by comparing the automatically parameterised model with the manually parameterised model from the publication.

Chapter 4 demonstrates the power of machine learning methods in building *Miscanthus* crop models. A wide range of machine learning algorithms and approaches are used and compared in their ability to produce accurate models. The models developed in this chapter are also validated against the automatically parameterised crop model from chapter 3.

Chapter 5 describes the use of the best performing machine learning methods from chapter 4, for the purposes of finding optimal phenotypic and environmental measurements for building accurate crop models. An optimisation algorithm is used to investigate ways of reducing the measurements in the model training data without negatively impacting modelling accuracy. The algorithm can be used to establish

importance of variables and time ranges of the growing season for developing accurate models and to inform future decisions for optimising data collection.

Chapter 6 discusses the answers to the scientific questions presented in section 1.5, the scientific significance of the work presented in this thesis and possible trends and topics for future research.

Chapter 2: Materials and methods

2.1 Data collection methods

2.1.1 Trial configuration

This section describes the configuration of all five trials I have collected data from (ABR 61, ABR 15, ABR 60, ABR 46 and HCK 1). At the start of the PhD project, 2 years of data from the ABR 61 trial were already available from the BBSRC Sustainable Bioenergy Centre (BSBEC) project. Thus, the main priority was to expand this dataset further. As the aim of the project was to optimise the phenotyping process, additional phenotypic data was collected from the ABR 15 and ABR 60 trials. In 2016, data was also collected from the ABR 46 and HCK 1 trials, which was intended for validation of the method developed in chapter 5.

Out of the five trials, only data from ABR 61 was used in the experiments described in chapters 3, 4 and 5. In ABR 15 and ABR 60 there were no dry weight measurements taken during the growing season, as this would have interfered with other scientific work in these trials. Time limitations prevented the use of the ABR 46 and HCK 1 data for the intended purpose, however, that validation was still done using the ABR 61 data. The description of these trials was included in this section, as the work done on them represented a significant portion of the data collection work done in this project.

ABR 61

The ABR 61 dataset was the most extensive dataset in this project. It was collected from the trial established by the BSBEC BioMASS Programme and funded by the BBSRC Sustainable Bioenergy Centre (BSBEC). It was planted in March 2009, located near Aberystwyth on the west coast of Wales (52.4142° N, -4.0433° W), and its soil type was classified as silty clay loam. It consisted of four genotypes: *M. sinensis* (EMI-11), *M. sinensis* (Goliath), *M. sacchariflorus* (Sac-5) and *M. x giganteus* (Gig-311) that exhibit a wide variation in canopy morphology. A randomised block design was used that had four replicate blocks each containing four plots, one for each genotype. Each

plot contained 121 plants (7.2 m x 8.5 m) at a density of $2\ \text{plants} \times \text{m}^{-2}$. Figure 2.1 contains a diagram of an example plot from the trial.

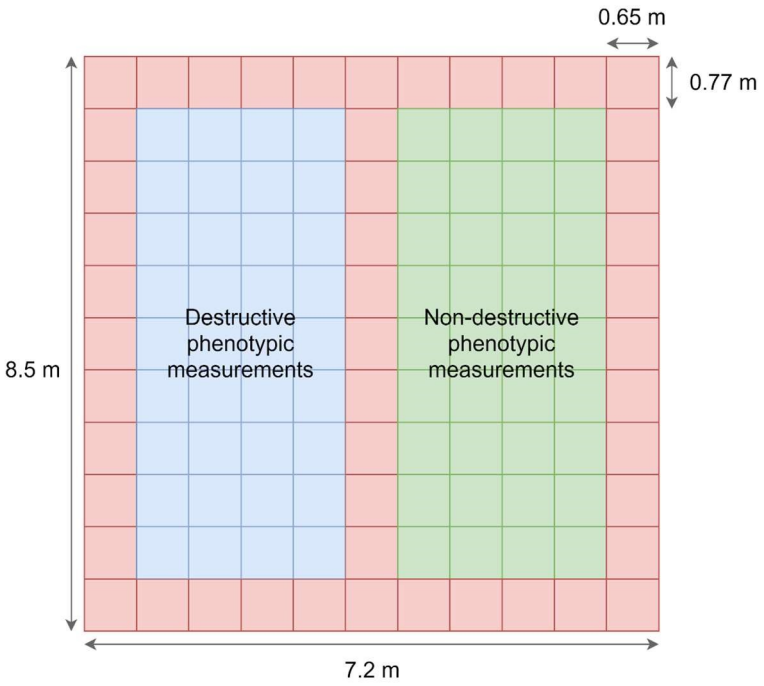

Figure 2.1 Diagram of a single plot from the ABR 61 trial. Each square is a single plant, which occupied an area with dimensions 0.65m by 0.77m. The red squares represent the border plants – they were not used for any measurements. The green squares are plants that were used for the collection of non-destructive phenotypic measurements. The blue squares denote plants that were used for destructive measurements.

Initially data from this trial were collected in the growing seasons of 2011 and 2012 as part of the BSBEC project (Davey *et al.*, 2015). For this thesis, I expanded the dataset by collecting data in 2014, 2015 and 2016.

In all years, except for 2016, non-destructive phenotypic measurements were made on a weekly or fortnightly basis. For these three plants were chosen at random from each plot in the beginning of each growing season. They were measured from late April until the beginning of October. In 2016 these measurements were taken once a month, from June until October (5 times in total). Phenotypic traits were averaged by plot, for example the average canopy height was taken from the individual measurements for each of the three plants of any given plot and assigned as the plot-level canopy height. Table 2.1 lists the traits measured in each year. Section 2.1.2 describes the phenotyping protocol for each trait. Plant emergence was recorded in

Growing season	Stem count	Canopy height	Stem diameter	Base diameters	Leaf area index	Flowering score	PAR transmission	Total destructive harvests	In-season destructive harvests	PAR	T_{max}	T_{min}	Rainfall
2011	✓	✓	×	×	✓	✓	✓	5	3	✓	✓	✓	✓
2012	✓	✓	×	×	✓	✓	✓	2	0	✓	✓	✓	✓
2014	✓	✓	×	×	✓	✓	✓	1	0	✓	✓	✓	✓
2015	✓	✓	×	×	✓	✓	✓	5	4	✓	✓	✓	✓
2016	✓	✓	✓	✓	✓	✓	✓	5	5	✓	✓	✓	✓

Table 2.1: Phenotypic and met data availability from ABR 61

2015, and its values were assumed to be the same for 2014 and 2016. The date for emergence was recorded by plot. A plot was considered to have emerged when two out of the three randomly chosen plants have emerged.

Destructive harvests were carried out several times during the growing season (between June and October) and once at final harvest time in February. In-season harvests have a special importance to training machine learning models because they give the ability to relate the increase in the amount of biomass to phenotypic changes in the plant (e.g. increase in stem count, canopy height, etc.) during the growing stage. For this reason, their number was increased in 2015 and 2016.

A meteorological station recorded air temperature, photosynthetically active radiation (PAR) levels, and rainfall hourly. These variables are described in more detail in section 2.1.3.

Data from nearby weather stations was used to fill any missing data. A daily degree days (DD) value was calculated using the maximum and minimum temperatures for each day, by applying McVicker's method of calculating degree days (McVicker, 1946). This method is described in detail in section 2.1.3.

The hourly measurements of PAR and rainfall were summed to daily ones. To match the daily values for PAR, rainfall and DD, with the less frequent phenotypic measurements, the daily sums of the meteorological variables were then

accumulated independently for each plot since its day of emergence. The accumulated values for each day where phenotypic measurements were recorded was taken to be the corresponding meteorological values. Table 2.2. contains a sample of the ABR61 data after the pre-processing step where the mean of the phenotypic measurements were calculated, and the matching cumulative meteorological values were inserted.

ABR 15

The trial was established in 2010 and was comprised of a population of 108 F1 plant hybrids resulting from a cross between *M. sacchariflorus* and *M. sinensis*. They were organised using a randomised block design, in three blocks, each containing every one of the 108 hybrid genotypes. The plants were planted on a grid, leaving 1.5m spaces between each one. The trial was located near Aberystwyth at the following coordinates: 52.437335° N, -4.026184° W, and its soil was dystric cambisol and dystric gleysol. There was no meteorological station in the same field as this trial and data from the ABR 61 meteorological station was used in the analyses for this thesis. I recorded non-destructive phenotypic measurements from the trial weekly during the growing seasons of years 2014 and 2015, starting from May and finishing in September. I collected destructive harvests data during the usual harvest period in February and March.

ABR 60

This trial was a comparison between three different planting densities; 50cm, 71cm and 100cm between each plant, and two different genotypes Goliath and Gig-311. It was established in 2009 and was located in the same field as ABR 61 and its soil was classified as silty clay loam. The randomised block design had three blocks each containing plots of the two genotypes planted at the three different densities. As this trial is in the same field as ABR 61 the same meteorological data was used for both.

PlotID	Genotype	Date	Stem count	Canopy height	Stem diameter	Base diameter 1	Base diameter 2	Leaf area index	Flowering score	PAR transmission	Dry weight	PAR	DD	Rainfall
48	EMI-11	28/06/2016	37	105	6	22	31	2.802	0	0.054	692.9	798.1	1187.2	160.9
48	EMI-11	12/07/2016	56	112	7	34	24	5.525	0	0.011	894.1	879.6	1378.8	225.6
48	EMI-11	09/08/2016	46	150	5	24	26	5.323	2	0.001	1931.1	1046.9	1816.8	267.5
48	EMI-11	07/09/2016	70	154	6	36	35	7.162	3	0.003	2497.3	1205.2	2278.9	331.7
48	EMI-11	04/10/2016	56	140	5	34	29	4.633	5	0.001	2015.2	1329.8	2656.7	403.3
56	Giganteus	28/06/2016	11	140	11	41	33	1.962	0	0.157	640.7	882.6	1285.8	169.6
56	Giganteus	12/07/2016	22	166	10	55	50	3.859	0	0.076	629.6	964.1	1477.4	234.3
56	Giganteus	09/08/2016	18	230	10	41	45	5.384	0	0.016	2042.0	1131.4	1915.5	276.2
56	Giganteus	07/09/2016	22	260	9	56	52	6.228	0	0.028	2189.8	1289.8	2377.6	340.4
56	Giganteus	04/10/2016	19	270	10	61	52	7.836	0	0.037	3082.4	1414.3	2755.4	412.0

Table 2.2 Sample from the ABR61 phenotypic and meteorological data. PlotID identified the plot where measurements were taken. PAR, DD and rainfall values were cumulative values since plot emergence. The phenotypic values shown in the table were the mean values for these measurements from the three chosen plants. The values for LAI, PAR transmission, dry weight, PAR, DD and rainfall presented in the table were rounded to conserve space.

ABR 46 and HCK 1

These trials were established in 2014 as part of GIANT-LINK, a public-private collaborative project between public partners DEFRA and BBSRC and industrial partners Ceres Inc., E.On, Blankney Estates, NFU and Terravesta (Clifton-Brown, Schwarz and Hastings, 2015). Some of the aims of the project included developing *Miscanthus* germplasm through breeding, evaluating the performance of new hybrids and improving agronomic practices for planting, managing and harvesting the crop. The trials belong to a group of six trials which were planted in different locations in the UK. The five genotypes (GNT 1, GNT 2, GTN 3, GNT 4 and GNT 5) included in all trials were produced through breeding in GIANT-LINK. ABR46 was located near Aberystwyth (52.4144° N, -4.0419° W), and its soil type was classified as silty clay loam. HCK 1 was located near Hackthorn, Lincolnshire (53.3319° N, -0.4708° W) and its soil type was loamy sand to sand.

As the main purpose of these trials was to test the commercial planting equipment replication was not a viable option in the trial design. Instead, the trials were laid out as non-replicated randomised blocks, one block for each genotype. Each block consisted of 52 plants per row, but the exact number of rows varied between trials. In HCK 1 it was 25 on average whereas in ABR46 it was 9. I collected both non-destructive and destructive phenotypic data from the two trials once a month from June until October in 2016.

2.1.2 Phenotyping methods

The following section describes the protocols used for obtaining phenotypic measurements.

Plant emergence

Miscanthus plants were harvested every year in February or March by cutting the above ground biomass at a height of approximately 10 cm. At the beginning of the following growing season, new shoots would emerge from the plant rhizome. Plant emergence was recorded as the date when the first leaf with a ligule was observed on any of the new shoots.

Stem count

For plants early in the growing season or small plants, the stem count was the total number of stems. However, later in the season there was a need to distinguish tall stems from young shoots that did not significantly contribute to the canopy or biomass yield. In this case, only stems taller than 60% of the canopy height were counted as described in (Clifton-Brown and Lewandowski, 2002).

Canopy height

Canopy height was measured as the distance from the base of the plant to the edge of the canopy. In planophile species like *M. sinensis*, this edge was the point where most of the leaves at the top of the plant were horizontal. In erectophiles like *M. sacchariflorus,* due to the upright stature of the leaves the canopy edge was taken as the middle of the leaves at the top of the plant.

Tallest stem

Along with canopy height, tallest stem is one of the traits that shows a high positive correlation with yield (Robson, Jensen, *et al.*, 2013). It is defined as the height of the tallest stem to the highest ligule.

Stem diameter

The stem diameter is the average stem width as measured at middle-height of the plant. Leaf nodes and stem nodules were avoided during the measurements. Between three and five diameters were recorded and then averaged to give the final trait value.

Base diameters

The area that all the plant's stems occupied was recorded by measuring two diameters at the base of the plant at 90 degrees to each other. If the plant base was highly elliptical the largest possible diameter was recorded by measuring the distance between the stems furthest apart.

Leaf width and length, and leaf area ratio

Leaf width and length were recorded for every leaf on a randomly chosen mature stem of average height. The area of each leaf (m^2) was estimated using the formula:

$$\text{leaf area} = \text{leaf width} * \text{leaf length} * \text{leaf area ratio}$$

The leaf area ratio in the formula is a coefficient that converts rectangular leaf area ($\text{leaf width} * \text{leaf length}$) (m^2) to actual leaf area. The value of the coefficient was calculated using a calibration data set, which included measurements of leaf width, length and leaf area of every intact leaf on average mature stems harvested during the growing season. The leaf area was estimated using image analysis. The formula for the leaf area ratio is:

$$\text{leaf area ratio} = \frac{\text{leaf width} * \text{leaf length}}{\text{estimated leaf area}}$$

The measurement unit of the leaf area ratio is thus $\text{m}^2 * \text{m}^{-2}$ or dimensionless.

Each genotype had a slightly different leaf shape, so potentially two leaves with the same length and width, coming from different genotypes would have had different leaf areas. Due to this variability in leaf shape, an individual leaf area ratio was calculated for each genotype.

Leaf area index

The leaf area index (LAI $\text{m}^2\ \text{leaf}\ \text{m}^{-2}\ \text{ground}$) was calculated using the formula:

$$\text{LAI} = \frac{\text{plant leaf area}}{\text{plant area}}$$

Where:

- plant leaf area (m^2) is the total leaf area of the plant
- plant area (m^2) is the area that each plant occupies on average and is determined by the planting density of the trial.

The plant leaf area was calculated using the formula:

$$\text{plant leaf area} = (\sum_{\text{i=1}}^{\text{n}} \text{leaf area}_\text{i}) * \text{stem count}$$

Where:

- leaf area_i (m) is the leaf area of leaf number i on the chosen stem
- stem count is the number of stems measured for the plant as defined above.
- $(\sum_{i=1}^{n} \text{leaf area}_i)$ the sum of all the leaf areas on the chosen stem approximates the average leaf area per stem. Multiplying that by the stem number gives the total plant leaf area.

Transmission

Transmission is the proportion of photosynthetically active radiation (PAR) not intercepted by the canopy and was measured using a SunScan SS1 (Delta T Devices Ltd, Cambridge, UK: www.delta-t.co.uk/). The instrument is composed of a data logger attached to a 1-meter long probe with 64 light sensors as well as a separate light sensor mounted on a tripod. The probe was placed at the bottom of the plant canopy and all 64 sensor readings recorded. The average of these values gave the $\text{PAR}_{\text{bottom}}$ value. The light sensor on the tripod was placed under direct sunlight, away from any shadow. It measured the amount of PAR that arrived at the top of the canopy, denoted by PAR_{top}. Whenever a $\text{PAR}_{\text{bottom}}$value was measured the SunScan SS1 also recorded the corresponding PAR_{top}. The PAR transmission was calculated using the following formula:

$$\text{PAR}_{\text{transmission}} = \frac{\text{PAR}_{\text{bottom}}}{\text{PAR}_{\text{top}}}$$

The result ranged from 0 to 1, where 0 denoted 0% of the light reached the bottom of the canopy (therefore 100% of the light was intercepted by the plant) and 1 denoted 100% of the light was reaching the canopy base (the plant was intercepting 0% of the light). Measurements with the SunScan SS1 were only taken between 1 hour before and after the solar noon, and never under rainy conditions as specified by the manufacturer's instructions. Two measurements were taken on each single plant at 90 degrees to each other. The average of the two $\text{PAR}_{\text{transmission}}$ values was used in the analysis. For the ABR61 dataset (described in section 2.1.3.) transmission was measured for three individual plants from each plot.

Flowering score

Flowering time is an important trait for yield as it marks the point in the growing season when the plant transitions to its reproductive phase, which in turn influences vegetative growth (Jensen *et al.*, 2011). Flowering was recorded using a six-point scoring system, which is also employed by the *Miscanthus* breeding programme at IBERS. The values range from 0-5 and their definitions are listed in Table 2.3.

Score	Definition
0	No flowering has occurred
1	Flag leaf emerged
2	Panicle emerged (>1cm)
3	Anthers emerged
4	Seed set
5	Flowering completed

Table 2.3: Flowering scoring system definition

Destructive harvest

In the field, the plant stems were tied and cut at a height of 10cm. The material was weighed using a system of tripod and hanging scales to give the plant fresh weight. Depending on the thickness of the stems, between 2 and 5 were chosen as a subsample and put in a paper bag with a label of the plant's unique identifier (UID). The subsample was weighed on laboratory scale yielding the subsample fresh weight value. It was then dried in an oven at 60°C until constant mass and then weighed again to give the subsample dry weight. The weight of the paper bag and label were subtracted from the results. Having measured the three values, the dry weight of the plant was calculated using the formula:

$$\text{plant dry weight} = \text{plant fresh weight} * \frac{\text{subsample dry weight}}{\text{subsample fresh weight}}$$

Occasionally, the amount of biomass produced by the plant was too small to be precisely measured with the hanging scales. In this case the whole plant was placed in the paper bag and weighed with the more precise laboratory scales. Thus, only plant fresh weight and plant dry weight values were recorded.

2.1.3 Meteorological data

Photosynthetically active radiation

The meteorological stations provided either hourly or half hourly average rates of PAR in $\mu\text{mol photons m}^{-2}\text{ s}^{-1}$. These values were converted to $\text{MJ m}^{-2}\text{ s}^{-1}$ using the formula below:

$$\text{PAR}_{\text{MJ}} = \frac{\text{PAR}_{\mu\text{mol}}}{4.55 * 10^{6}}$$

The constant 4.55 converts the μmol to J and 10^{6} convert the J to MJ. The rate values were then converted to quantities of PAR per hour (or half hour, depending on the logging frequency of the meteorological station) and the result values from each day were summed to give total daily values of PAR. The conversion is described in the following formula:

$$\text{PAR}_{\text{daily}} = \sum \text{PAR}_{\text{MJ}} * \text{logging frequency} * 60$$

Logging frequency was either 30 or 60 depending on whether the meteorological station records values every 30 or 60 minutes. $\text{PAR}_{\text{daily}}$ had units of MJ m^{-2}. When training a model, PAR was provided as the cumulative daily PAR values since the plant emerged.

Degree days

Degree days were calculated daily using the maximum and minimum daily temperatures. The McVicker's (McVicker, 1946) method was used as it is the standard for calculating degree days in the UK (CIBSE TM41, 2006). There are four cases with different formulas, that depend on the daily values for maximum and minimum temperature and how they compare to the base temperature. The conditions and the corresponding formulas are listed in Table 2.4.

Case	Condition	Formula
1	$T_{max} \leq T_b$	$T_b - \frac{(T_{max} + T_{min})}{2}$
2	$T_{min} < T_b$ and $(T_{max} - T_b) < (T_b - T_{min})$	$\frac{(T_b - T_{min})}{2} - \frac{(T_{max} - T_{b})}{4}$
3	$T_{max} > T_b$ and $(T_{max} - T_b) > (T_b - T_{min})$	$\frac{T_b - T_{min}}{4}$
4	$T_{min} \geq T_b$	0

Table 2.4: Degree day formulae for different cases

Where:

- T_{min} (°C) is the minimum daily temperature
- T_{max} (°C) is the maximum daily temperature
- T_b is the base temperature for *Miscanthus*, which is different for each genotype. The base temperature is the minimum temperature, below which no development or growth would occur in the plant. As part of the work described in this thesis was to replicate the model described by (Davey *et al.*, 2015), the same approach of assuming T_b to be 0°C for all genotypes was taken.

In this thesis, the degree day values used were the cumulative degree days since plant emergence.

Rainfall

The weather stations measured rainfall in millimetres, giving hourly or half-hourly totals. These were summed to daily values. For the purposes of modelling rainfall was converted to the cumulative daily values since plant emergence.

2.2 Machine learning

Data analysis is one of the most important tools of science. It allows us to form and test hypotheses on observed data and draw conclusions based on the results. Analysis often involves a modelling approach where statistical or machine learning methods are employed to identify relationships and patterns in the data. The goal of machine learning is the development of computer algorithms that can automatically extract knowledge from data which in turn improves their performance at a certain

task (Mitchell, 1997). Machine learning problems can broadly be classified into two categories: supervised learning and unsupervised learning.

2.2.1 Supervised and unsupervised learning

In supervised learning, the algorithm is provided with a dataset $\mathbf{Z}$ of N number of examples (instances) $\mathbf{x}$ coupled with corresponding label (response) $\mathbf{y}$ like so $\mathbf{Z} = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), (\mathbf{x}_3, \mathbf{y}_3), \dots (\mathbf{x}_\mathrm{N}, \mathbf{y}_\mathrm{N})\}$. Each instance consists of a set of common features X. A feature is a measurable property of the instance that may or may not influence the value of the label. The label is the target variable of interest that provides useful information for the instance. The goal is to predict the correct value of y for a given $\mathbf{x}$. The algorithm uses the dataset to build or "train" a prediction model (learner) that uses $\mathbf{x}$ as input and outputs the correct y. A useful analogy is that supervised learning is similar to a teacher (user) providing training examples to the student (algorithm).

By way of contrast, the dataset in unsupervised learning problems consists solely of unlabelled input instances. The goal is to identify some underlying hidden structure or probability density function of the data. A popular method is clustering where instances are grouped together based on similarity. While in supervised learning there is a direct measurement of success by comparing predicted to actual values, the nature of unsupervised learning problems does not allow for the same techniques, as there is no available "answer" to compare to (Hastie, Tibshirani and Friedman, 2009).

The rest of this section focuses on algorithms and methods for supervised learning which were used in the research described in this thesis.

2.2.2 Supervised learning algorithms notation

Sections 2.2.5 and 2.2.6 contain descriptions of various supervised machine learning algorithms. They all assume the presence of a training dataset $\mathbf{Z}$ of N number of instances and responses: $\mathbf{Z} = \{(\mathbf{x_1}, \mathbf{y_1}), (\mathbf{x_2}, \mathbf{y_2}), (\mathbf{x_3}, \mathbf{y_3}), \dots (\mathbf{x_N}, \mathbf{y_N})\}$, as described in section 2.2.1. Each instance $\mathbf{x}$ is a vector made up of p number of features: $\mathbf{x} = \{\mathrm{X}_1, \dots, \mathrm{X}_p\}$. Each individual feature X is a single value. The response $\mathbf{y_1}, \mathbf{y_2}$, etc. could contain multiple features $\mathbf{y} = \{\mathrm{Y}_1, \dots, \mathrm{Y}_\mathrm{K}\}$. However, in most cases it is assumed that

they contain a single value, in which case vector notation will not be used and the response will be denoted as $\mathrm{y}_1, \mathrm{y}_2$.

The vector of instances from the training dataset $\mathbf{Z}$ denoted by $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, \dots, \mathbf{x_N}\}$ is referred to as the input data, as this is the data that a machine learning model takes in, to produce predictions. The corresponding set of labels $\mathbf{Y} = \{\mathrm{y}_1, \mathrm{y}_2, \dots, \mathrm{y}_\mathrm{N}\}$ is called the output data, as this is the desired output from the model. The model predictions, which are the labels that the model actually outputs, $\widehat{\mathbf{Y}} = \{\hat{\mathrm{y}}_1, \hat{\mathrm{y}}_2, \dots, \hat{\mathrm{y}}_\mathrm{N}\}$ are distinguished from the training output data $\mathbf{Y}$ using the caret symbol.

A machine learning model is a function f which given a single input instance $\mathbf{x}$ produces a single output label $\hat{\mathrm{y}}$:

$$\hat{\mathrm{y}} = f(\mathbf{x})$$

Similar notation is used when the input is all the input data $\mathbf{X}$ and the output is all the model predictions $\widehat{\mathbf{Y}}$**:**

$$\widehat{\mathbf{Y}} = f(\mathbf{X})$$

2.2.3 Regression and classification tasks

The first step in data analysis is building models of the data using machine learning methods and training strategies. Depending on the type of analysis that needs to be performed, the tasks in supervised learning are either classification or regression. Classification requires the algorithm to learn and predict categorical target variables. In comparison, the target variable type for regression is continuous. Most supervised learning algorithms can perform both tasks. However, some of them are limited to just one. ID3, for example, can only perform classification (Quinlan, 1986), while logistic regression is designed to predict binary categorical variables (Kurt, Ture and Kurum, 2008). The choice of algorithm is thus determined by the required task. But for each task there is a wide variety of algorithms available, each with its own approach to modelling training data. The modelling strategies have their individual strengths and weaknesses which make them perform differently depending on the problem they are applied to. No single algorithm models every dataset better than all the rest.

2.2.4 Bias-variance trade off, underfitting and overfitting

An important metric of how algorithms perform is their prediction error. The prediction error is a function which calculates how much predictions deviate from target values. It can be divided into three components (Hastie, Tibshirani and Friedman, 2009):

$$\mathrm{Err}(\mathrm{x_i}) = \text{Irreducible error} + \text{Bias} + \text{Variance}$$

The irreducible error is the naturally occurring variation of the target variable around its mean. There is nothing that could be done to reduce that component. Bias denotes the model bias – how much does the expected prediction of the model drift from the target. Variance is the deviation of the expected prediction for a single data point around its mean. An overly simple model would fail to capture the nature of the relationship between inputs and response, which would lead to high bias. This is known as underfitting. Increasing the model complexity decreases the bias, but a too complex model has high variance. It would be performing well only on the training dataset but given another sample from the same distribution it would lose prediction accuracy. This is the problem of overfitting - the model has tried to fit so closely to the training data that is has failed to generalise to unseen data. Models are prone to underfitting and overfitting to a different extent, depending on their design and strategies to avoid these problems.

2.2.5 Parametric methods

The list of methods begins with parametric algorithms. At their basis lies the assumption that the training samples are drawn independently from a probability distribution defined by a model (Alpaydin, 2010). The model is trained by estimating its parameters from the training examples. Once the parameters are known, the model can estimate the full distribution.

Linear regression

Linear regression is the most widely used method of regression and is the backbone of statistics (Hastie, Tibshirani and Friedman, 2009). For a given instance defined by a vector of p features $\mathbf{x} = \{\mathrm{X_1}, \mathrm{X_2}, \mathrm{X_3} \ldots \mathrm{X_p}\}$ and response variable y, the linear regression model is:

$$y = \beta_0 + \sum_{i=1}^{p} X_i \beta_i + \varepsilon$$

Where:

- X_i is the i-th feature of instance $\mathbf{x}$
- β_0 is the intercept, also called the bias parameter. It enables the model to account for any fixed offset in the data.
- β_i is the parameter for the i-th feature. The model "learns" by finding the optimal values for the $\boldsymbol{\beta}$ parameters.
- ε is the error term. It is a zero mean Gaussian random variable, which accounts the variability in the data that is not explained by the input features.

As the input consists of a set of n instances $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3} \dots \mathbf{x_N}\}$ and a vector of response variables $\mathbf{Y} = \{y_1, y_2, y_3 \dots y_N\}$ the formula above can be generalised to the whole training dataset using matrix notation:

$$\mathbf{Y} = \mathbf{X}^T \boldsymbol{\beta} + \varepsilon$$

Where:

- $\boldsymbol{\beta}$ is a vector of all parameters including the intercept β_0, of length $p + 1$
- $\mathbf{X}$ is an $(N \times (p + 1))$ matrix. A constant feature $X_0 = 1$ has been added to each instance, to make it possible to calculate the inner product $\mathbf{X}^T \boldsymbol{\beta}$.

There are many methods to fit the parameters to the data, but the most popular is the method of least squares. It calculates the values of the $\boldsymbol{\beta}$ parameters that minimise an error function. An error function is a function that uses the deviation of the prediction from the real response to evaluate the model accuracy. In the case of least squares, it is defined as the sum of the squared difference between predicted and actual values. This function is called the residual sum of squares.

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{N} (y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2$$

The estimated parameters $\widehat{\boldsymbol{\beta}}$ are calculated using the following formula:

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}(\mathbf{X}^{\mathrm{T}}\mathbf{Y})$$

Finally, the model can be used to compute the fitted values $\widehat{\mathbf{Y}}$ using the formula:

$$\widehat{\mathbf{Y}} = \mathbf{X}^{\mathrm{T}}\widehat{\boldsymbol{\beta}}$$

The predictive success of linear regression depends on the nature of the data. For example the method is known to have limitations when applied to problems with high-dimension input spaces (Bishop, 2006). There are several assumptions that linear regression makes, which if untrue, would result in poor model performance. One such assumption is multicollinearity of the features, i.e. they are independent of each other. If this is not true, it affects feature coefficients (Mansfield and Helms, 1982) which can lead to dubious results. In that case, it is better to select a minimum set of features to build a model on, using subset selection methods like best-subset selection, forward- and backward-stepwise selection, or forward-stagewise regression (Hastie, Tibshirani and Friedman, 2009). Subset selection either discards or keeps a variable, which can lead to models with high variance and therefore fail to improve predictive accuracy. The solution to this are shrinkage methods like ridge regression and LASSO (Hoerl and Kennard, 1970; Tibshirani, 1996) which look for a solution that minimises the least squares and the values of the β parameters. The parameters are shrunk towards zero which yields a model with lower variance. Linear regression was used for training models in the work described by chapters 4 and 5. All linear regression models were trained using the "lm" function in the R programming language (R Core Team, 2017).

Artificial neural networks

The name for neural networks comes from the fact that they were first developed to model the human brain (Hastie, Tibshirani and Friedman, 2009). There are a large variety of methods that fall into the neural network group among which the one hidden layer perceptron, described here, is the most commonly used. It consists of input features $\mathbf{x} = \{\mathrm{X}_1, \mathrm{X}_2, \mathrm{X}_3 \ldots \mathrm{X}_\mathrm{p}\}$, a hidden layer of M derived features or neurons $\mathbf{z} = \{\mathrm{Z}_1, \mathrm{Z}_2, \mathrm{Z}_3 \ldots \mathrm{Z}_\mathrm{M}\}$ and output vector $\mathbf{y} = \{\mathrm{Y}_1, \mathrm{Y}_2, \mathrm{Y}_3 \ldots \mathrm{Y}_\mathrm{K}\}$. Its structure is shown on Figure 2.2.

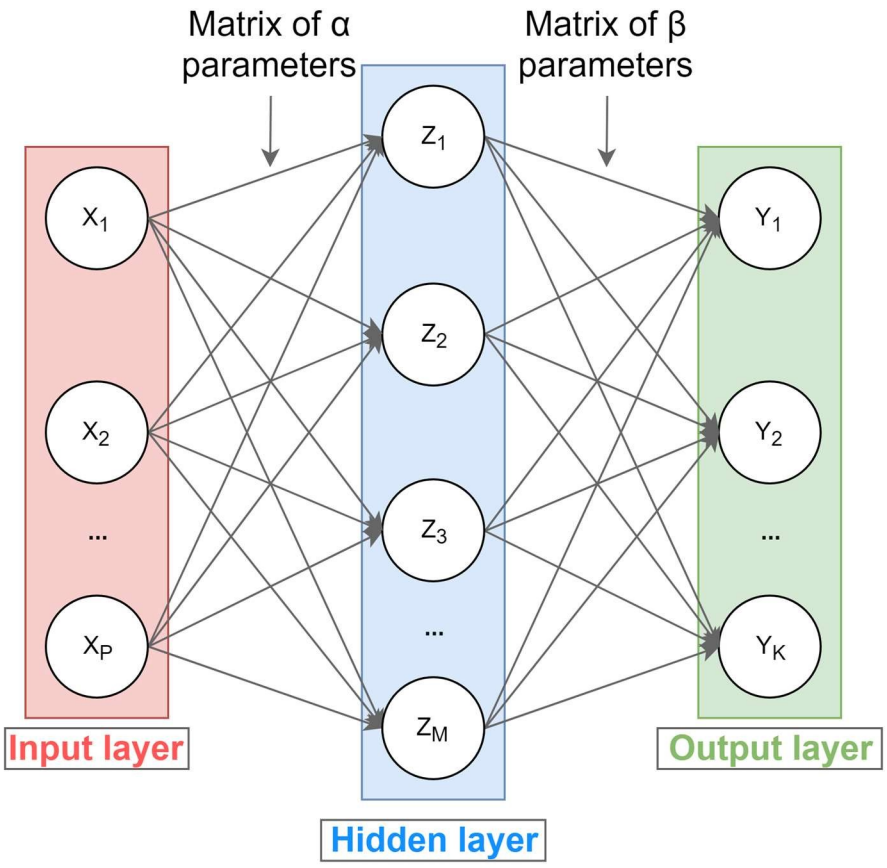

Figure 2.2 Structure of a one hidden layer neural network. The input layer consists of the input features $\boldsymbol{x} = \{X_1, X_2, X_3 \dots X_p\}$*, the hidden layer has* M *derived features or neurons* $\boldsymbol{z} = \{Z_1, Z_2, Z_3 \dots Z_M\}$ *and the output layer contains the output vector* $\boldsymbol{y} = \{Y_1, Y_2, Y_3 \dots Y_K\}$*. The lines between the features are the* α *and* β *parameters described later in this section.*

It is important to note that the assumption for the response variable $\mathbf{y}$ containing a single value, as stated in section 2.2.2 is not true for the section of the text which describes neural networks – this was done to demonstrate a more general use case of neural networks, where their output consists of multiple features $\hat{\boldsymbol{y}} = \{Y_1, Y_2, \dots, Y_K\}$.

The hidden features are derived from the input features using the formula:

$$Z_m = \sigma(\alpha_{0m} + \boldsymbol{\alpha}_m^T \mathbf{x})$$

Where:

- m takes values from 1 to M
- σ is the activation function for the derived features. It determines whether the neuron will fire, i.e. whether and how much it will influence the final prediction.
- α_{0m} is a bias parameter or weight individual for each Z_m. It has a similar use to the intercept in linear regression. $\boldsymbol{\alpha}_0$ is a vector of length M containing all bias parameters.
- $\boldsymbol{\alpha}_m$ is a vector of length p containing weights for each input feature. $\boldsymbol{\alpha}$ is an $(M \text{ x } p)$ matrix of weights.

The activation function is usually the sigmoid function:

$$\sigma(v) = \frac{1}{(1 + e^{-v})}$$

It produces an s-shaped curve and returns values between 0 and 1. After the features of the hidden layer are derived, they are used to calculate the outputs:

$$\hat{Y}_k = \beta_{0k} + \boldsymbol{\beta}_k^T \mathbf{z}$$

Where:

- k takes values from 1 to K
- β_{0k} is the bias weight for output $\hat{Y}_k$. $\boldsymbol{\beta}_0$ is a vector of length K that contains all bias parameters.
- $\boldsymbol{\beta}_k$ is a vector of length K containing weights for each neuron. $\boldsymbol{\beta}$ is a $(K \times M)$ matrix of weights.

It is particularly evident by the last equation that neural networks greatly resemble linear regression. It has been argued that they are nonlinear generalisations of linear regression due to the nonlinearity of the activation function (Bishop, 2006; Hastie, Tibshirani and Friedman, 2009).

Training the neural network requires finding the optimal values for the weights that minimise the sum-of-squares error of prediction. For a given training dataset with input $\mathbf{X} = \{x_1, x_2, x_3 \dots x_N\}$ and output $\mathbf{Y} = \{\mathbf{y_1}, \mathbf{y_2}, \mathbf{y_3} \dots \mathbf{y_N}\}$ the sum-of-squares error is:

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(\mathbf{x}_i))^2$$

Where:

- y_{ik} is the kth value of output i
- $f_k(\mathbf{x}_i)$ is the kth value output from the neural network
- θ is the complete set of weights

One way to minimise the sum-of-squares is backpropagation. It uses gradient descent to modify weights until a minimum is reached. A minimum is a solution where no further improvement is gained when modifying the weights. The algorithm goes through multiple iterations updating the weights each time, until it reaches the solution. If the error is represented as the sum of errors for each individual training example:

$$R(\theta) = \sum_{i=1}^{N} R_i$$

then the weights at iteration $(r + 1)$ are:

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}}$$

$$\alpha_{km}^{(r+1)} = \alpha_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{km}^{(r)}}$$

Where:

- $\alpha_{km}^{(r)}$ and $\beta_{km}^{(r)}$ are weight values from iteration r
- γ_r is the learning rate

The learning rate could be a constant or could be made to decrease toward zero as $r \to \infty$. A large learning rate offers rapid learning but it could prevent the algorithm from reaching an optimal solution, causing it to oscillate between suboptimal ones. With a small learning rate, it is more likely to get stuck in local minima, points in the solution space where neighbouring solutions are not better, but globally these solutions are poor. Due to the small learning rate, the algorithm would never change the weights as it will be looking at the gradient over a small distance.

Artificial neural networks were used for training models in chapter 4, using the nnet function in R (Venables and Ripley, 2002).

Deep learning

Most machine learning models have a restricted capacity for directly using input data in its raw form and often require its conversion into a vector of features (LeCun, Bengio and Hinton, 2015). Deep learning methods build multiple hierarchically dependent layers, following a similar structure to an artificial neural network with many hidden layers. Raw data is passed through the first layer, which transforms it into a more abstract representation, which is then passed as input to the next layer. This process is repeated sequentially at each layer, resulting in higher-levels of representation of the data each time. For example, in an image processing problem,

the first layer might detect edges in the image from raw pixels, the second would detect corners and contours from the output of the previous layer (edges), and the process continues upwards in the hierarchy until the final layer detects the object in the image (Goodfellow, Bengio and Courville, 2016).

It was established that backpropagation did not perform well in assigning weights to each layer of a deep learning model, because of its tendency to get struck in local minima which was exacerbated when the number of hidden layers increased (Yu and Deng, 2011). This hampered the development of deep learning, until 2006 when an unsupervised learning algorithm was developed for optimising the initial weights for a class of deep learning neural networks, called deep belief networks (DBN) (Hinton, Osindero and Teh, 2006). The weight values were assigned in a greedy manner, which reduced the time complexity of the algorithm. After initialising the weights in this way, backpropagation was used to fine tune their values, which gave better results than backpropagation over randomly initialised weights.

A number of different deep learning architectures have been applied to a wide variety of problems in signal processing, like speech recognition, computer vision and language and text processing (Yu and Deng, 2011), while it has also found use outside computer science, in computational biology (Angermueller *et al.*, 2016) and biomedicine (Mamoshina *et al.*, 2016). The increasing size of training datasets as well as advances in computer hardware have been crucial for the success of deep learning. Deep learning models require large amounts of data, which was not historically available and this impacted their performance in the past (Goodfellow, Bengio and Courville, 2016). The advances in computer hardware since the 1980s has resulted in the increase in model sizes, and the ability of deep learning methods to take advantage of faster CPUs, GPUs, and improved distributed computing has enabled them to build very accurate models. Deep learning was not used in this thesis, because the training data collected during this project was not large enough for it to be successful.

2.2.6 Nonparametric methods

While parametric models define a single model for the whole feature space, nonparametric methods divide it into regions based on similarities between the training examples. Then they fit a different model to each region. (Alpaydin, 2010).

K-nearest neighbour

K-nearest neighbour takes a different approach to most algorithms, as it does not create a model of the data. Instead, the data acts as the model and the predictions provided are based directly on the training examples. It was initially developed for classification (Fix and Hodges, 1989). Given a training dataset (**X**, **Y**) as defined in section 2.2.2, and a new query point x_0, k-nearest neighbour follows the steps:

- Calculate the distance between x_0 and all other training examples. Assuming the features are real numbers, Euclidean distance is the most popular choice, but other options exist depending on the feature space.
- Pick the k nearest training examples $\mathrm{D} = \{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3} \ldots \mathbf{x_k}\}$ to $\mathbf{x}_0$.
- The predicted class of x_0 is simply the majority vote of the examples in D.

The method is easily modified for regression problems. The predicted output $\hat{\mathrm{y}}$ is simply the average of the response variables in D (Farahnakian *et al.*, 2013).

$$\hat{\mathrm{y}} = \frac{\sum_{\mathrm{i}=1}^{\mathrm{k}} \mathrm{y_i}}{\mathrm{k}}$$

The choice for the parameter k greatly influences the performance of the algorithm. A large k would reduce the effect of noisy data but would introduce bias in the predictions. Cross-validation can be used to check the prediction error associated with each k value. Cross-validation is a technique where a small part of the dataset is withheld to test the performance of the learner on unseen data. Section 2.1.6 provides a more detailed explanation of the different approaches for cross-validation.

A problem in standard k-nearest neighbour arises with the assumption that the distance between x_0 and the training examples in D is relatively small (Denoeux, 1995). However, that is not always the case, which leads to questionable predictions, if examples that are further away are allowed to participate with the same weight on

their vote as the ones nearer to the query point. This is further exacerbated if there is an imbalance in the distribution of classes, i.e. the majority of training examples belong to just one of the classes. A solution is to use a modified version of k-nearest neighbours that adds weights to the training examples (Dudani, 1976). This allows the nearer training examples to have more influence over the output.

As is the case for linear regression, k-nearest neighbour has limitations with high-dimensional feature space problems. This is due to the nearest examples being more likely to be further away from the query point, which impacts negatively the performance of the learner (Hastie, Tibshirani and Friedman, 2009). In this thesis, k-nearest neighbour models were trained in chapters 4 and 5, using the kknn function in R (Schliep and Hechenbichler, 2016).

Tree-based methods

Tree-based methods use decision trees as a model for data. Binary decision trees are generally preferred (Street, 2005), and will be the focus of this text. The root and branch nodes on the tree represent an inequality test between a feature value and a constant (greater than, smaller than, etc.). They partition the feature space into M rectangular regions $\mathrm{R} = \{\mathrm{R}_1, \mathrm{R}_2, \mathrm{R}_3, \ldots \mathrm{R}_\mathrm{M}\}$. Instances are sorted through the tree according to the value of their features until they reach a leaf node. A leaf node represents one of the regions in the feature space.

A separate model is fit to each region and is applied to all instances that fall within it. Usually the model outputs a single constant value c_m for a region R_m. Thus, the decision tree can be expressed as:

$$\mathrm{f}(\mathbf{x}) = \sum_{\mathrm{m}=1}^{\mathrm{M}} \mathrm{c}_\mathrm{m}[\mathbf{x} \in \mathrm{R}_\mathrm{m}]$$

Where:

- $\mathbf{x} \in \mathrm{R}_\mathrm{m}$ is 1 if $\mathbf{x}$ belongs to region R_m and 0 otherwise

The constant value is often set as the mean output value for the examples in region R_m.

$$\mathrm{c}_\mathrm{m} = \mathrm{ave}(\mathrm{y}_\mathrm{i} | \mathbf{x}_\mathrm{i} \in \mathrm{R}_\mathrm{m})$$

The algorithm starts training by first partitioning the feature space in two regions. The new regions are defined by the variable j and value s on which the algorithm splits.

$$R_1(j, s) = \{\mathbf{x} | x_j \leq s\}$$

$$R_2(j, s) = \{\mathbf{x} | x_j > s\}$$

The aim is to choose values for j and s that minimise the prediction error in the two regions. As it is computationally infeasible to find the best split using minimum squares a greedy algorithm is used (Hastie, Tibshirani and Friedman, 2009).

$$\min_{j,s}[\min_{c_1} \sum_{\mathbf{x} \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{\mathbf{x} \in R_2(j,s)} (y_i - c_2)^2]$$

After finding the best split, the algorithm moves on to further partition the regions recursively following the same rules, until a stopping condition is met. The final regions are the leaves on the tree.

Bootstrapping, bagging and random forests

Decision trees have low bias, but high variance. They are sensitive to small perturbations in the training data, which can lead to drastically different trees. This makes decision trees unstable (Breiman, 1998; Binongo *et al.*, 2007). The variance could be reduced by bagging, where multiple versions of the learner are used for prediction. This is the technique employed by ensemble methods like random forests, which is a modification of decision trees that has been shown to greatly improve prediction accuracy (Breiman, 2001a; Caruana and Niculescu-Mizil, 2006).

Training multiple decision trees requires different versions of the training dataset. To achieve this, random forests uses bootstrapping. Assuming a dataset Z of size N, new datasets Z^* of the same size are formed by drawing with replacement from Z. As they are drawn with replacement, they contain approximately a random 63.2% of the samples in the original (Alpaydin, 2010). This is repeated B times, depending on the number of trees that are to be trained. Finally, the algorithm returns the ensemble of trees $\mathbf{T} = \{T_1, T_2, \dots, T_B\}$. Random forests predict a new instance x by taking the mean of the individual tree predictions.

$$f(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} T_b(\mathbf{x})$$

Bootstrapping and bagging are applicable to other algorithms as well. There have been multiple studies where they were successfully applied to neural networks, linear regression, support vector machines, naïve Bayes learner and k-nearest neighbour (Ha, Cho and Maclachlan, 2005; Braga *et al.*, 2007; Liang and Zhang, 2010). In this thesis, the random forests algorithm was used for training models in chapters 4 and 5, using the randomForest function in R (Liaw and Wiener, 2002)

Support vector machine

Support vector machines (SVM) were first developed as a classification algorithm (Cortes and Vapnik, 1995). The basic concept is to fit a hyperplane that best separates two sets of examples belonging to different classes. It is then used to classify new examples, depending on which side of the hyperplane they lie.

A modification of SVM called support vector regression (SVR) was later proposed (Drucker *et al.*, 1997), which had the ability to work on regression problems. Its mechanism is explained in detail by Smola and Schölkopf (Smola, Sch and Schölkopf, 2004) and only parts of it will be covered here. In its simplest form SVR fits a hyperplane defined by a linear equation to the data:

$$f(\mathbf{x}) = \mathbf{x}^T\boldsymbol{\beta} + \beta_0$$

Where:

- As was the case in parametric models, $\boldsymbol{\beta}$ is a vector of feature weights
- β_0 is an intercept term

The algorithm aims to find a function $f(\mathbf{x})$ that predicts all training examples with deviation from the response y of up to ε, i.e. all training examples fall within the area between two margin hyperplanes $f(x) + \varepsilon$ and $f(x) - \varepsilon$. At the same time, it is required that the model is as flat as possible. Flatness (complexity) means small β. The result is an optimisation problem:

$$\text{minimise } \frac{1}{2}\|\boldsymbol{\beta}\|^2$$

$$\text{subject to } \begin{cases} y_i - \mathbf{x}_i^T\boldsymbol{\beta} - \beta_0 \leq \varepsilon \\ \mathbf{x}_i^T\boldsymbol{\beta} + \beta_0 - y_i \leq \varepsilon \end{cases} for\ every\ i = 1, \dots, N$$

It is assumed that there exists such a hyperplane that could fit all training examples within the error margin ε. As this is not always the case, a soft margin is used which deals with the training examples with deviation larger than ε. Two slack variables ξ_i, ξ_i^* are introduced, which represent the deviation of each example respectively below and above the margins of the linear function following the rules:

$$\xi_i^* = \begin{cases} 0 \text{ if } \xi \geq \varepsilon \\ |\xi| - \varepsilon \text{ otherwise} \end{cases}$$

$$\xi_i = \begin{cases} 0 \text{ if } \xi \leq -\varepsilon \\ |\xi| - \varepsilon \text{ otherwise} \end{cases}$$

$\xi = y_i - \mathbf{x}_i^T\boldsymbol{\beta} + \beta_0$ is the deviation of the training example from the predicted value. A positive value means the training example is situated above the line and negative means it is below.

This leads to the updated optimisation problem:

$$\text{minimise } \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$$

$$\text{subject to } \begin{cases} y_i - \mathbf{x}_i^T\boldsymbol{\beta} - \beta_0 \leq \varepsilon + \xi_i \\ \mathbf{x}_i^T\boldsymbol{\beta} + \beta_0 - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \\ C > 0 \end{cases}$$

The second term added to the function being minimised (the objective function) is a penalty term for the soft margin. The constant C determines the trade-off between large soft margin and model complexity. The optimisation problem is solved using the method of Lagrange multipliers. A Lagrange function L is built using the objective function and the constraints. Its minimisation with respect to the parameters $\boldsymbol{\beta}$ and ε yields the dual optimisation problem:

$$\text{maximise} \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ -\varepsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{N} y_i(\alpha_i - \alpha_i^*) \end{cases}$$

$$\text{subject to } \sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \text{ and } \alpha_i, \alpha_i^* \in [0, C]$$

Where:

- α_i, α_i^* are parameters of the Lagrange function

Finding the values for α_i, α_i^* make it possible to calculate β using:

$$\boldsymbol{\beta} = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)\mathbf{x}_i$$

The method described up to now fits a linear function. However, if the relationship between input and output is not linear that would be a poor model of the data. Support vector machines make use of the principle of basis expansion, where each input vector x_i is mapped to a higher dimension feature space using a function h. An example function h assuming $\mathbf{x}_i = \{X_1, X_2\}$ could be $h(\mathbf{x}) = \{X_1, \sqrt{2 * X_1 * X_2}, X_2\}$. Fitting the same linear function $f(\mathbf{x}) = h(\mathbf{x})^T\beta + \beta_0$ to the expanded feature space results in a curve in the original feature space.

SVM builds up on basis expansion by exploiting the fact that the optimisation function requires the dot product of input vectors $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$. A kernel function k is defined where $k(\mathbf{x}, \mathbf{x}') = \langle h(\mathbf{x}), h(\mathbf{x}') \rangle$. If the result of $k(\mathbf{x}, \mathbf{x}')$ is known, there is no need to compute $h(\mathbf{x})$. The kernel function replaces the dot product of the input vectors in the first equation of the optimisation problem:

$$-\frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(\mathbf{x}_i, \mathbf{x}_j)$$

Kernels make it possible to use very large, possibly infinite dimensional enlarged space, which would otherwise be computationally infeasible (Hastie, Tibshirani and Friedman, 2009). Popular kernels for SVM are:

- dth-Degree polynomial $\mathrm{k}(\mathbf{x}, \mathbf{x}') = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^{\mathrm{d}}$
- Radial basis $\mathrm{k}(\mathbf{x}, \mathbf{x}') = \mathrm{e}^{-\gamma \|\mathbf{x} - \mathbf{x}'\|^2}$
- Neural network $\mathrm{k}(\mathbf{x}, \mathbf{x}') = \tanh(\mathrm{k}_1 \langle \mathbf{x}, \mathbf{x}' \rangle + \mathrm{k}_2)$

In this thesis, SVM models were trained as part of the work described in chapters 4 and 5, using the LiblineaR function in R (Helleputte, 2017).

Boosting methods

Much like bagging, boosting combines multiple weak learners to produce a prediction. It differs in the training procedure of the models. This can be illustrated with a description of the most popular boosting algorithm Adaboost.M1 (Freund and Schapire, 1997; Hastie, Tibshirani and Friedman, 2009) which is designed for classification problems.

Each input vector is first assigned an importance weight $\mathrm{w} \in [0,1]$. Initially the input vectors are set to be equally important $\mathrm{w_i} = \frac{1}{\mathrm{N}}, \mathrm{i} = 1,2, \ldots \mathrm{N}$. A weak classifier G_1 is trained on the data. The weights of the misclassified examples are increased while the weights of the correctly classified ones are decreased. A weight α_1 is assigned to G_1 depending on its prediction error. A second weak classifier G_2 is trained on the reweighted data. Its error function includes the weights, so the learner is forced to prioritise on correctly classifying the misclassified instances. Afterwards the examples are reweighted, α_2 is assigned, and the process is repeated until M learners are trained. The prediction of the boosted model is a weighted majority vote of the M learners.

An algorithm, called gradient boosting, was later developed which extended the boosting approach to regression problems (Friedman, 2001). Like Adaboost it trains sequentially a set of weak learners called base learners. While gradient boosting works with a wide range of base learners, decision trees are usually used. Unlike bagging models, here the model is a sum over all the base learners' predictions.

Assuming $\mathrm{f}(\mathbf{x})$ is the model, h is the base learner and $\boldsymbol{\theta}_\mathrm{m}$ are the parameters for h in the m-th iteration.

$$\mathrm{f}(\mathbf{x}) = \sum_{\mathrm{m}=1}^{\mathrm{M}} \mathrm{h}(\mathbf{x}, \boldsymbol{\theta}_\mathrm{m})$$

Gradient boosting trains its base learners in iterations. In the first iteration, the learner is trained on the dataset pairs $(\mathbf{x_1}, \mathrm{y}_1), (\mathbf{x_2}, \mathrm{y}_2), \ldots, (\mathbf{x_N}, \mathrm{y}_\mathrm{N})$ like normal. Each next learner aims to minimise a loss function L. The loss function is a function that takes predicted and actual values $\mathrm{L}(\mathrm{y}, \mathrm{f}(\mathbf{x}))$ and its value increases the more predicted values deviate from actual values. An example of loss function is least squares which is used in linear regression.

On each iteration m, gradient boosting looks for the best set of parameters $\boldsymbol{\theta}_\mathrm{m}$ for h, such that when the result of the function is added to the model from the previous iteration $\mathrm{f}_{\mathrm{m}-1}$would minimise prediction error.

$$\boldsymbol{\theta}_\mathrm{m} = \mathrm{argmin}_{\boldsymbol{\theta}_\mathrm{m}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{L}(\mathrm{y}_\mathrm{i}, \mathrm{f}_{\mathrm{m}-1}(\mathbf{x}_\mathrm{i}) + \mathrm{h}(\mathbf{x}_\mathrm{i}, \boldsymbol{\theta}_\mathrm{m}))$$

$argmin$ denotes the value for $\boldsymbol{\theta}_m$ that minimises the loss function L. The new base learner $\mathrm{h}(\boldsymbol{\theta}_\mathrm{m}, \mathbf{x})$ is chosen to be parallel to the negative gradient of the loss function. This allows the algorithm to avoid looking for a general solution and instead employ a greedy strategy which greatly simplifies computation (Natekin and Knoll, 2013). The negative gradient is:

$$\mathrm{g}_\mathrm{m}(\mathbf{x}_\mathrm{i}) = -\left[\frac{\partial \mathrm{L}(\mathrm{y}_\mathrm{i}, \mathrm{f}_{\mathrm{m}-1}(\mathbf{x}_\mathrm{i}))}{\partial \mathrm{f}_{\mathrm{m}-1}(\mathbf{x}_\mathrm{i})}\right]$$

The gradient parameter selection rule differs depending on the loss function. In the case of least squares, the rule is:

$$\theta_\mathrm{m} = \mathrm{argmin}_{\theta_\mathrm{m}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \left(-\mathrm{g}_\mathrm{m}(\mathbf{x}_\mathrm{i}) - \mathrm{h}(\mathbf{x}_\mathrm{i}, \boldsymbol{\theta}_\mathrm{m})\right)^2$$

As is evident from the formula the base learner's target is no longer the output values $\{y_1, y_2, \dots, y_N\}$ but rather the negative gradient, so the training dataset in iteration m is in fact:

$$\{(g_m(\mathbf{x}_1), \mathbf{x}_1), (g_m(\mathbf{x}_2), \mathbf{x}_2), \dots, (g_m(\mathbf{x}_N), \mathbf{x}_N)\}$$

The model is updated with the new base learner:

$$f_m(\mathbf{x}) \leftarrow f_{m-1}(\mathbf{x}) + h(\mathbf{x}, \boldsymbol{\theta}_m)$$

Training stops when the model $f_m(\mathbf{x})$ consists of M base learners.

Boosting has the advantage of training very accurate models. However, this comes at the price of slow training time (De'ath, 2007; Caruana, Karampatziakis and Yessenalina, 2008). An implementation of gradient boosting termed "Generalised Boosted Model" (GBM) (Ridgeway, 2015) was used to train models as part of the work described in chapter 4. This was done in R using the gbm function (Ridgeway, 2015).

2.2.7 Model validation

The second step of data analysis is the evaluation of model accuracy and selection of the model that fits the data best. This is done based on test error, a metric that measures the deviation of predicted responses from real values, on an independent test sample τ, using an arbitrary loss function L.

$$\mathrm{Err}_\tau = E[L(y, f(\mathbf{x}))|\tau]$$

Where:

- E is the expected value of the loss function

Some approaches, like residual plot and goodness of fit use the training dataset directly to measure model performance. Residual plots produce a graph that could aid in determining whether the relationship between the input variables and the response is linear. Goodness of fit is a group of measurements that define how well the model fits the data. The most commonly used one is the coefficient of determination R^2. The drawback of these methods is that usually they use the training error to estimate the test error. However, it is also possible to calculate

goodness of fit over an independent test sample, which is the recommended approach (Alexander, Tropsha and Winkler, 2015). There are multiple different formulas used in literature for calculating R^2 (Kvålseth, 1985), and the one used in this thesis is the squared Pearson's correlation coefficient (Ahlgren, Jarneving and Rousseau, 2003):

$$R^2 = \left(\frac{\sum_{i=1}^{n}(z_i - \bar{z}) \sum_{i=1}^{n}(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(z_i - \bar{z})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \right)^2$$

Where:

- $z_i = f(\mathbf{x}_i)$ is the model prediction for $\mathbf{x}_i$
- $\bar{z}$ is the average of all model predictions
- $\bar{y}$ is the average of all response values $\mathbf{Y}$

More generally the training error can be defined as the average loss over the training dataset.

$$\mathrm{Err}_{\mathrm{training}} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i))$$

Common options for the loss function are mean square error (MSE), root mean square error (RMSE), mean absolute error (MAE) and median absolute error (MdAE) (Hyndman and Koehler, 2006). Besides R^2, modelling accuracies are compared using RMSE in this thesis:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2}$$

Training error provides a poor estimation of the test error (Hastie, Tibshirani and Friedman, 2009). As mentioned earlier an overfitting model would perform well over the training dataset, but fail to predict unseen data. One example of this is the 1-nearest neighbour model. Its error would appear to be 0 as it will be returning the exact same values it was given on training. However, this learning strategy results in high variance as the model's predictions are greatly influenced by the noise in the individual training instances. This increases the true error of the model, but it cannot

be detected using the training dataset. A widely used strategy is leaving a small part of the dataset for testing purposes, which simulates the model predicting an independently drawn dataset. It can be used to identify overfitting models, as they would have low training error but high estimated test error.

A method for estimating test error in cases where data are not plentiful is k-fold cross-validation. It starts by splitting the dataset into K equal parts, using sampling without replacement. A set of K models is trained on the data, each time leaving one of the parts out. A model trained without part k is denoted by $\mathrm{f}^{-\mathrm{k}}(\mathbf{x})$. The test error is estimated by calculating the expected loss from predicting each part k using $\mathrm{f}^{-\mathrm{k}}(\mathbf{x})$.

$$\mathrm{CV}(\mathrm{f}) = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{L}(\mathrm{y}_\mathrm{i}, \mathrm{f}^{-\mathrm{k}}(\mathbf{x}_\mathrm{i}))$$

The choice of K is an important decision. If $\mathrm{K} = \mathrm{N}$, which is also known as leave-one-out cross-validation, the test error estimate is unbiased, and the computation cost is the highest possible. The downside is that the estimate can have high variance and therefore provide unreliable results (Kohavi, 1995; Hastie, Tibshirani and Friedman, 2009). Increasing K lowers the variance but increases the bias and computation cost. The choice depends on the size of the dataset, and usually five- or tenfold cross-validation are recommended. Cross-validation is an effective method to estimate test error in small datasets, where splitting the data into training and testing partitions is not practical. However, ideally, an independent dataset would be used to validate the model.

Bootstrap can also be used for test error estimation. The simplest way is to train a set of models on B bootstrap samples. Then the error is estimated over the whole dataset. However, the instances used for training overlap some of the instances used for testing. This introduces bias which is almost as high as the one present in estimations of test error using training error. The ".632 estimator" is an improved approach which adds the principle of cross-validation. The estimated error is a combination of the error over the full dataset E_b and the error calculated only on the instances outside the bootstrap sample $\varepsilon 0$.

$$E_{boot} = \frac{1}{B}\sum_{i=1}^{B}(0.632\varepsilon 0_i + 0.368E_b)$$

Where:

- $\varepsilon 0_i$ is the prediction error for instance i or the average error on that instance of all models which have not seen it during training.

E_b tends to underestimate the true error, whereas $\varepsilon 0$ tends to overestimate, so using them both is supposed to bring the estimated error close to the true error. However, this method has been shown to fail with "perfect memoriser" learners like one nearest neighbour. The training data for such models could be manipulated to move the error estimation in any direction (Kohavi, 1995).

Cross-validation and bootstrap can be used to prevent the model from overfitting. They can be used alongside the training and test dataset splitting strategy by applying them to the training partition. Optimal parameters are chosen based on their error estimate, and the model is validated against the test examples.

2.2.8 Machine learning and statistics

Machine learning is not the only way to model data. In his paper "Statistical Modelling: The Two Cultures" (Breiman, 2001b) Breiman compares two general approaches to modelling: data modelling and algorithmic modelling. They both view data as being generated by a black box, where an unknown process associates input variables x to output variables y. The two goals of modelling are to predict outputs for future inputs and gain insight into the process inside the black box.

Data modelling is the established approach among statisticians. Data is assumed to be generated from independent draws from a known model. Conclusions about the process mechanism are based on the model's mechanism. Algorithmic modelling originates from computer science and artificial intelligence. It employs algorithms which aim to emulate the outcome of the process inside the black box as closely as possible.

The two approaches focus on two different aspects of modelling, which predetermines their choice of models. Data modelling aims to explain the process

behind existing data and use modelling to test hypothesis of causal relationship between the input variables X and the response Y (Shmueli, 2010). Models are chosen with their interpretability in mind. Algorithmic modelling concentrates on producing a model with high predictive power. The attention is on properties and training strategies of algorithms which enable them to produce good predictors.

Model validation

Depending on whether the goal is explanatory or predictive modelling, different techniques are used in each stage of the analysis (Shmueli, 2010). A good example is the disparity in model validation. Data modelling relies on goodness of fit tests and residual analysis to determine the accuracy of its models. As mentioned before, the use of the training dataset for accuracy estimation yields unrealistically good predictive accuracy. In this context, the methods aim to determine explanatory power rather than predictive power over the existing data. This may be justified so long as it is not assumed that high explanatory power means high predictive power.

These methods are not perfect even when used as intended. It has been demonstrated that goodness of fit tests lack power and fail to reject linearity. In high dimensional datasets, the interaction between variables prevents residual analysis from detecting lack of fit (Breiman, 2001b). Breiman argues for using cross-validation and test datasets, as the lack of predictive accuracy in a model signals its failure to emulate the process inside the box. Therefore, any conclusions drawn from such a model would be questionable.

Model selection

A common problem of both data and algorithmic models is the multiplicity of good models. There are often multiple models that perform equally well according to the error rate estimate. In explanatory modelling, this presents a huge issue for interpreting results. For example, there could be multiple linear regression models built on different sets of variables with different parameters, that fit equally well to the data. Each one lists a different set of variables as the most important, presenting several competing, but equally likely hypotheses. With no difference in their error, there is no way of telling which one is closest to the truth.

Predictive modelling uses this to its advantage. Ensemble methods rely on training multiple unstable learners that have low bias and high variance (Breiman, 1998). Aggregating the predictions over all learners deals with the variance problem, gives a significantly more accurate model than using any single one of them, and increases the uniqueness of the solution (Breiman, 2001b).

Model complexity

The need for interpretability in data modelling limits the complexity of the models. This is why linear regression is the preferred approach for regression problems (Shmueli, 2010). Models like random forest and neural networks are often great predictors, but their inner workings are complex and cannot be examined in the same way as linear models. That does not mean that machine learning algorithms are useless in explaining tasks. With the right approach, useful information can be extracted from these models. Breiman demonstrated this by using variable importance scores from random forests to form new hypotheses (Breiman, 2001b). Arguably, the information coming from a complex predictive model is likely to be more reliable due to its superior predictive accuracy. This is true, as long as the model is not too complex, which would cause it to overfit its training data.

Curse of dimensionality

A limitation of data models is their inability to train over high-dimensional data, also called the “curse of dimensionality”. The standard approach is to reduce dimensions by either removing variables or combining variables into fewer and more informative features. The approach in algorithmic modelling is the complete opposite – where possible increase the number of features, because the higher the number of features, the more information there is in the dataset. Newer machine learning models perform better the more variables they are given (Breiman, 2001b). This allows them to discover aspects of the data that remain hidden to standard models.

2.3 Crop modelling

Crop models are important tools for synthesising scientific knowledge of genetic, physiological and environmental processes, and their interaction in crops. They can be used in research for hypothesis testing and integrating knowledge across

disciplines, and decision making in agricultural management and policy making (Boote, Jones and Pickering, 1996). Crop models consist of mathematical equations that describe behaviour in plants and their interaction with the environment. They mimic plant growth by simulating the behaviour of its parts: roots, stems, leaves, flowers (Jame and Cutforth, 1996; Oteng-Darko *et al.*, 2013).

2.3.1 Empirical and mechanistic models

There is a variety of yield modelling methods based on their approach to modelling, purpose and design, which can be grouped into empirical and mechanistic. Empirical models aim to provide accurate crop yield predictions, to inform agricultural management. They consist of empirical functions that can be easily parameterised which model yield as a direct result of one or several observed variables. They do not model the internal mechanism of the plant and this allows them to achieve better prediction accuracy, as it reduces the uncertainty of parameterising the model (Di Paola, Valentini and Santini, 2016). This however means, that they don't contain any information on the underlying biological mechanism (Oteng-Darko *et al.*, 2013). For example, the yield comparison study between *M. x giganteus* and switchgrass (*Panicum virgatum)* modelled yield response to temperature, precipitation and N fertiliser using linear regression (Heaton, Voigt and Long, 2004). The models developed in this study were empirical models – they do not contain any information on the internal mechanism of the plants, but rather were functions that fit the observed data well, and modelled yield as a direct result of external factors.

Mechanistic process models (also referred to as process models) simulate plant growth in terms of the established knowledge of underlying physical, chemical and biological processes. Each component of a mechanistic model could describe a single process occurring in the plant that has been studied in isolation, like single-leaf photosynthesis, plant respiration, transpiration, etc. Models simulate these intermediate processes, their interactions and contribution to yield to predict the final crop production and thus form a more complete picture of plant growth and development. They can be used to explain the reasons and internal mechanisms behind growth responses and are thus an important research tool for expanding

biological knowledge. They can also highlight important areas that lack information and understanding (Boote, Jones and Pickering, 1996; Sinclair and Seligman, 1996).

In most cases crop models are based on a mixture between the mechanistic and empirical approaches (Monteith, 1996), as it allows models to benefit both from the scientific understanding of the processes as well as the information contained in empirical data.

2.3.2 Crop model parameterisation and validation

Often, the model predictions do not exactly fit the experimental data which was used to construct the model. Model parameterisation (calibration) refers to the process of modifying the model structure and finding the best parameter values that enable the model to produce accurate predictions for the data (Oteng-Darko *et al.*, 2013). This is equivalent to the training procedure in machine learning models. Once parameterised a crop model will need to be validated.

Validation establishes the ability of the model to predict outside of the data that it has already been parameterised on, in a way similar to machine learning model validation described in section 2.2.6. The crop model is tested against independent datasets and its prediction accuracy is calculated. This determines its ability to generalise to different sites and environmental conditions (Boote, Jones and Pickering, 1996). Validating the ability of a model to correctly predict yield or intermediate processes in real world observations data is an important method of testing the hypotheses that underlie it.

Crop models are covered in more detail in chapter 3, where an automated parameterisation procedure is developed for a *Miscanthus* growth model (Davey *et al.*, 2015), which modelled growth by simulating the total leaf area of the plant and the amount of light intercepted by it.

2.4 Evolutionary optimisation

Optimisation problems are present everywhere – in engineering, industry, business there is a constant search for ways of performing tasks with minimum effort and

maximum gain (Yang, 1970). Optimisation is the area of computational science which seeks to solve such problems though the means of computer algorithms.

The optimisation problems encountered during the work described in this thesis, were tackled using a type of optimisation algorithms called evolutionary optimisation algorithms. Two evolutionary algorithms are described in this thesis – in chapter 4 genetic algorithms (GAs) were used to find the optimal configuration for a compound model made up of several machine learning submodels. The GAs were used to identify the optimal arrangement of the submodels, to maximise their accuracy for predicting *Miscanthus* yield. In chapter 5, another evolutionary algorithm – scatter search, was used to find an optimal solution to the cost-benefit problem of providing optimal training dataset for *Miscanthus* machine learning models predicting yield.

2.4.1 Genetic algorithms

GAs are a class of algorithms that have been inspired by the principles of biological evolution (Srinivas and Patnaik, 1994). While the exact procedure is specific to each implementation, they all work on a randomly generated set of solutions to the problem (population), which they recombine to produce better solutions, while discarding the poor solutions from the population. This seeks to emulate the process of evolution in nature.

Figure 2.3 illustrates an example optimisation problem – a stick character needs to get as close as possible to the green square in four moves. At each turn, the character moves one square and the possible moves are UP, DOWN, LEFT, RIGHT, but the character cannot move onto a black square. In GAs and other evolutionary optimisation algorithms, solutions are encoded as sets of decision variables, that

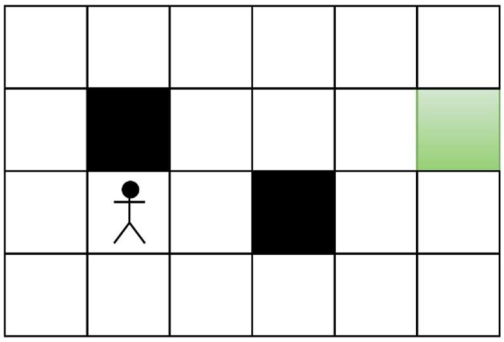

Figure 2.3 Example optimisation problem - the stick character needs to get as close as possible to the green square in four moves. At each move the character can move one square up, down, left or right, unless the square it is moving into is black. Optimisation algorithms can be used to search for the best sequence of moves for this problem.

describe the solution. For example, the optimal solution to the problem in Figure 2.3 could be encoded in four variables x_1, x_2, x_3, x_4, the value of which describe respectively the first, second, third of fourth move:

$$x_1 = RIGHT$$
$$x_2 = UP$$
$$x_3 = RIGHT$$
$$x_4 = RIGHT$$

Each of these variables can only take one of the possible move values. In GAs the decision variables are referred to as genes, and the whole solution is called a chromosome. The solutions are evolved through several strategies: selection, recombination and mutation. Selection is the process of retaining the most beneficial (or best) individual solutions from a population, which is an implementation of the survival of the fittest principle in biological evolution. Recombination strategies are methods of creating new solutions by combining genes from two existing solutions,

mimicking chromosomal crossover. In the optimisation problem illustrated above, an example recombination between two solutions would be taking the value for x_1 and x_2 from the first and the value for x_3 and x_4 from the second solution. Mutation is the process of randomly changing a single value on a solution chromosome, similar to substitution mutation in DNA. Example of this would be changing the value of x_4 from RIGHT to a randomly chosen move (e.g. UP).

The decision on the benefit from each solution lies in the utility function. This is a function that must be specifically defined for each problem and its output is the utility value of a given solution. That value will denote how beneficial the solution is, and depending on the nature of the problem we will be seeking to either maximise or minimise the utility function. For example, if the algorithm seeks to maximise sales for a business, the utility value would reflect the sales figure, and the algorithm will be searching for solutions that maximise the utility function. But in the example in Figure 2.3, the utility value could be the distance between the character and the green square, which we seek to minimise. In that case, solutions with low utility value would be better.

The genetic algorithm applies the three strategies to get a new set (generation) of solutions, which then becomes the new population. This process is repeated over and over until the algorithm reaches a stopping condition, such as finding an optimal solution, reaching a desired time limit or failing to find better solutions. Figure 2.4 illustrates the iterative process, as it was implemented in chapter 4. To create a new generation the algorithm:

- directly selected a number of the best solutions from the population and kept them for the next generation
- recombined randomly selected solutions from the population. Better solutions had a better chance of being selected for recombination. Some of the recombined solutions were also mutated – this was also determined randomly. The resulting solutions were kept for the next generation.

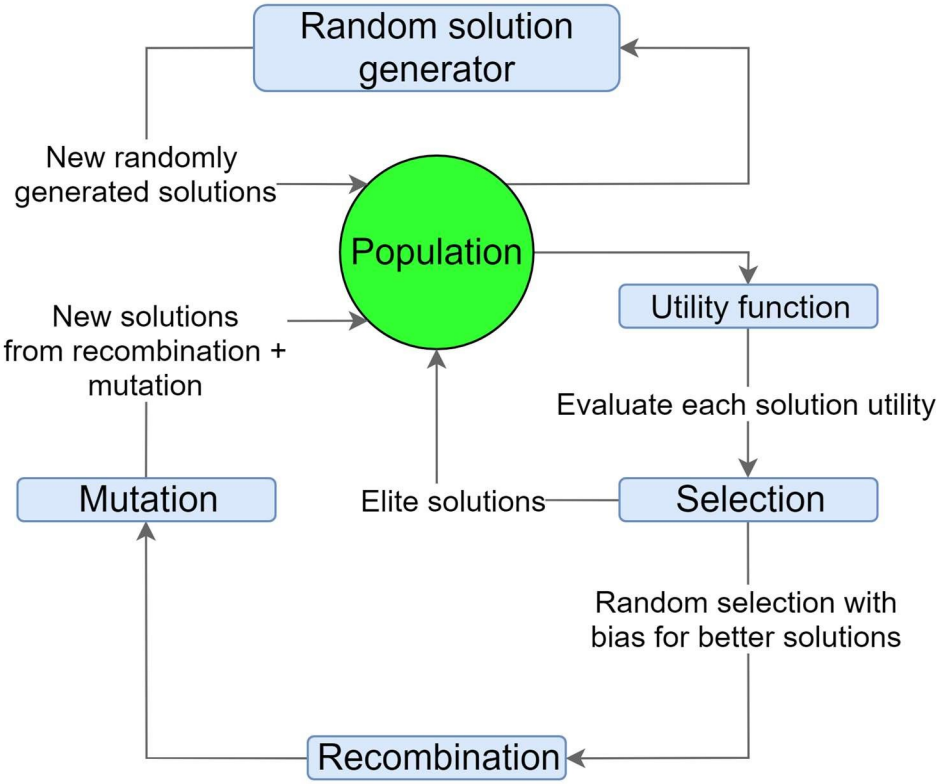

Figure 2.4 Flowchart of each iteration of the Genetic Algorithm, as it was implemented in chapter 4. The algorithm started with a population of randomly generated solutions. Their utility value was calculated for each solution. A number of the best solutions were kept for the following generation (elite solutions). Solutions were randomly selected for recombination, with bias towards the better solutions. Some of the resulting solutions were mutated. All the recombined solutions were kept for the next generation. Finally, a number of randomly generated solutions were also inserted in the new generation. The new generation of solutions replaced the solutions in the population, and the process was repeated until a stopping condition was met.

- a number of randomly generated solutions was also inserted into the next generation – this kept the algorithm from getting stuck in a local minimum and allowed it to explore a wider area of the search space.

The exact implementation of this genetic algorithm is described in more detail in section 4.2.4.

2.4.2 Scatter Search algorithms

Scatter search algorithms were first introduced in 1977 (Glover, 1977) as a heuristic for integer programming problems, which are optimisation problems where the decision variables can take integer values (Schrijver, 1986). Being evolutionary algorithms, scatter search methods generate and improve a set of solutions which is referred to as the reference set. They differ from genetic algorithms in that they aim to populate the reference set with both beneficial but also diverse solutions. Also, the solution improvement strategy in genetic algorithms consists of randomly changing and combining parts of solutions to create improved ones, while scatter search systematically explores all options to improve the solutions in the reference set. The size of the reference set is defined by the user and tends to be smaller than the genetic algorithms populations (Martí, Laguna and Glover, 2006).

As with genetic algorithms, there is no single version of scatter search, as algorithms are redesigned and adapted to the exact problem they are applied to. However, they usually follow the Scatter Search template (Glover, 1998; Martí, 2006) which lists methods that the algorithm applies to find optimal solutions. The precise implementation of these methods depends on the problem and the specific algorithm they belong to:

- Diversification generator –generates a diverse set of solutions
- Improvement method – manipulates solutions to create new and improved solutions
- Reference set update method – update the reference set with the best and most diverse solutions found so far
- Subset generation method – obtains a subset from the reference set of solutions, to be recombined with other solutions

- Solution combination method – combines the subset of solutions from the subset generation method into new solutions

According to the template, scatter search consists of two phases – the initial phase and the scatter search phase. The initial phase generates a set of good and diverse solutions by following two steps:

- Generate a diverse solution (from the ones currently in the reference set) using the diversification generator
- Improve the solution as much as possible by applying the improvement method

The generated solutions are inserted into the reference set and the process is repeated until the reference set is fully populated with solutions. The scatter search is then applied repeatedly to improve the solutions in the reference set. It consists of the following steps:

- Take a subset from the reference set using the subset generation method
- Combine solutions from the subset with the solution combination method
- Improve the resulting solutions and update the reference set with them with the improvement and reference set update methods

The steps are repeated until the solutions in the reference set cannot be improved anymore.

The scatter search algorithm used in this project was applied to a binary optimisation problem – there the decision variables took binary values. The algorithm and its implementation is described in sections 5.2.4, 5.2.5, 5.2.6 and 5.2.7.

2.5 Software

Chapters 3, 4, and 5 describe various models, model training, parameterisation and validation procedures, and other algorithms, which in this section will be referred to as “modules” for brevity. Most modules have been implemented in the Python 2.7 programming language (Python Software Foundation, 2013), unless stated otherwise. Notable exceptions include machine learning algorithms described in section 2.1. The implementation of these came from the following packages nnet,

LiblineaR, gbm, kknn, randomForest and caret of the R programming language (Liaw and Wiener, 2002; Venables and Ripley, 2002; Ridgeway, 2015; Kuhn, 2016; Schliep and Hechenbichler, 2016; Helleputte, 2017; R Core Team, 2017). Some modules, have not been mentioned in the following chapters, as they play a supporting role and are of no scientific interest. These include the data module, responsible for accessing and combining the phenotypic and meteorological data described in section 2.2 and organising it into a standard data record format. A Python software infrastructure was sitting above all modules, and was responsible for utilising the right set of modules, to implement the experiments described in chapters 3, 4, and 5. The logical flow of operations for each experiment was usually: load the data using the data module, pass that data to another module that trains and validates a model, and output the results.

The data plots presented in the following chapters were made using the ggplot2 R package (Wickham, 2009). Flowcharts and other diagrams were created using the graphviz software package, Dia diagram editor or the online diagram software Draw.io (www.draw.io) (Gansner and North, 2000; Breit *et al.*, 2007). The python and R code developed during this thesis are hosted at an online repository on this URL: https://github.com/yosifovemil/thesis/ .

Chapter 3: Automated procedure for *Miscanthus* process model training

This chapter describes the development of an automated procedure for parameterisation of a previously published mechanistic process model of *Miscanthus* yield referred to as the BSBEC process model (BPM) (Davey *et al.*, 2015). This resulted in a new model called the automatically parameterised BSBEC process model (APBPM). APBPM was trained using the same data as BPM, and their simulations on the training data and on an independent dataset were compared. APBPM produced similar predictions to BPM, demonstrating the ability of the automated procedure to generate a model equivalent to BPM. This meant that APBPM could be parameterised against an arbitrary dataset which was done in later chapters. The need for automation brought slight improvements in some of APBPM's modules, which improved the model accuracy, particularly in predicting late season leaf area index (LAI). The wider implications of this work are that APBPM can easily be parameterised for a new genotype, if it is provided with the necessary training data for that genotype.

3.1 Introduction

Crop models are a tool for simulating plant development and growth, with a wide range of applications in research, agriculture and policy making (Boote, Jones and Pickering, 1996; van Ittersum *et al.*, 2003). They can provide deeper understanding of the interaction between genotype, environment and crop management (Craufurd *et al.*, 2013). Example applications include estimation of crop yield potential in various geographical regions, climate change influence over crop performance, identifying appropriate crop management practices (Boote, Jones and Pickering, 1996), and possible environmental consequences of adopting a crop.

During the last 20 years, a considerable number of growth models have been developed for *Miscanthus*. They usually are based on the principles described by (Monteith and Moss, 1977), where the amount of dry biomass is expressed as a function of the available photosynthetically active radiation (PAR), proportion of light

intercepted by the plant (Pl), and the efficiency of conversion of radiation into biomass (radiation use efficiency or RUE) (Clifton-Brown *et al.*, 2000):

$$\text{yield} = \text{PAR} * \text{Pl} * \text{RUE}$$

One basic model was described by (Clifton-Brown *et al.*, 2000). It used degree days to simulate LAI, which was then used to calculate the light interception. RUE was calculated using a regression of dry matter yield on the intercepted PAR. The model was parameterised on *Miscanthus x giganteus* and was used to calculate yield potential for the whole of Ireland using a GIS. Another model built on the same principle (Price *et al.*, 2004) was also parameterised on *M. x giganteus* and was used to estimate the yield across England and Wales. Its advantage is that it also considered water availability and its influence over growth.

A subsequent paper described an improved version of the model (Clifton-Brown *et al.*, 2000), which became known as MISCANMOD (Clifton-Brown, Stampfl and Jones, 2004). The changes were related to estimation of the end of growing season using flowering time and degree days, and simulation of the influence of soil water availability to the plant. MISCANFOR, an updated and more detailed version of MISCANMOD, brought substantial improvements and new features to the model (Hastings *et al.*, 2009). It enhanced the description of light interception, and the influence of water stress and temperature on growth. It also modelled phases of growth, drought and frost kill, and nutrient repartitioning between above and below ground biomass. The improvements in MISCANFOR significantly increased its predictive power over that of MISCANMOD. Both MISCANMOD and MISCANFOR were parameterised for *M. x Giganteus* and were used to calculate the yield potential of that genotype across Europe. More recently, MISCANFOR was used to estimate the future potential yields of *Miscanthus* in Europe, assuming more chilling tolerant cultivars are developed through breeding (Kandel *et al.*, 2016).

The two models require substantial amounts of high resolution data for parameterisation and simulation, which is not always available. To tackle this problem a simplified version was made that was designed to work with monthly

meteorological data (Pogson, 2011). It reduced the computational time and made model parameterisation easier.

An extended version of the model in (Clifton-Brown *et al.*, 2000) different from MISCANMOD was described by (Farrell *et al.*, 2006). It used emergence and frost tolerance data to regulate the simulated LAI growth depending on environmental factors and genotype. The model was parameterised for four different genotypes, and was used to evaluate the importance of early emergence and frost tolerance to increasing the amount of biomass while keeping its interannual variability low. This was reinforced by a simplified version of MISCANMOD, which was parameterised for four genotypes, that were specifically chosen for their diverse canopy architecture (Davey *et al.*, 2015). In this thesis, the model is referred to as the BSBEC process model (BPM). It demonstrated that forming a canopy earlier in the season has more potential to increase yield than changing the canopy architecture to increase light interception (k). Another model developed on the same principles as MISCANMOD and MISCANFOR (Monteith and Moss, 1977) identified further parameters for optimising yield (Pallipparambil, Raghu and Wiedenmann, 2015). Solar radiation was found to have the strongest positive relationship with yield, while LAI, rainfall and plant available water had a limited influence, only in their lower ranges. Water loss, loss of leaves prior harvest and evapotranspiration were shown to have a strong negative influence over yield. Harvest date was shown to be important as well, with the optimal date for harvest being December 15, and earlier or later harvests impact yield negatively.

Other models have been developed by parameterising, adapting or otherwise extending already existing crop models. For example, the model created by (Miguez *et al.*, 2009) is an adaptation of the generic plant production model WIMOVAC (Windows Intuitive Model of Vegetation response to Atmosphere and Climate Change) (Humphries and Long, 1995), which was parameterised for *M. x giganteus*. The approach is based on simulating photosynthesis on the leaf and canopy level, growth partitioning, plant respiration, and water potentials in the plant and soil. Another approach described in (Stričević *et al.*, 2015) involves parameterising the FAO AquaCrop model (Raes *et al.*, 2009) for *Miscanthus*. The AquaCrop model differs

from the models described above, as it did not consider radiation and temperature but rather modelled yield as a response to water availability (Steduto *et al.*, 2009). This makes AquaCrop generic enough to be applicable to any plant species provided that enough training data is provided. Another generic model LINTUL(Light INTerception and UtiLisation simulator) (Spitters, 1987) served as the basis for the crop model LINPAC(LINTUL model for Perennial and Annual Crops) (Jing *et al.*, 2012). LINTUL was expanded with descriptions of LAI, light use efficiency (LUE) and base temperature, and parameterised for four genera: *Miscanthus* (*Miscanthus spp.*), reed canary grass (*Phalaris arundinacea L.*), willow (*Salix spp.*), and eucalyptus (*Eucalyptus spp.*). The three models (Miguez *et al.*, 2009; Jing *et al.*, 2012; Stričević *et al.*, 2015) provided frameworks for predicting biomass yield potential in a variety of geographical locations.

The potential of *Miscanthus* to reduce pollutants in water prompted a study into the effects of its cultivation on riverine nitrate load (Ng *et al.*, 2010). A model was created in the Soil and Water Assessment Tool (SWAT) (Arnold and Fohrer, 2005), simulating above and below ground biomass, LAI and nutrient stress. The model can be used to estimate the amount of land necessary to be cultivated with *Miscanthus* for reducing nitrate down to a certain target. The results suggested that *Miscanthus* is a favourable candidate crop for producing energy and reducing nitrate at the same time. A later study modified the research version of the crop-soil model STICS to better predict *M. x giganteus* data (Strullu *et al.*, 2014). The simulations from the model showed that *M. x giganteus* growth is limited by water and nitrogen availability during establishment and only by water availability in later seasons (Strullu *et al.*, 2015). The model could be used to estimate the impacts of *Miscanthus* cultivation on water drainage and nitrogen leaching.

The main objective of this chapter is to describe the development of an automated method for the parameterisation of BPM, resulting in a new model called the automatically parameterised BSBEC process model (APBPM). APBPM was trained using the same data as BPM, then validated against it, and tested using real data, demonstrating the ability of the automated procedure to parameterise a model which replicates the simulations that can be done by BPM. More importantly, the

new model could be easily parameterised against an arbitrary dataset. This was used in later chapters, where APBPM was trained using new datasets, and datasets with reduced number of data points. Because of this, the automation procedure was designed to be robust against perturbations in the training data.

3.2 Materials and methods

The procedure described in this section is largely based upon the approach described in (Davey *et al.*, 2015). However, some of the steps could not be reproduced directly and were modified to allow automation. For example, BPM assigned common parameter values between genotypes, if the difference between the individual genotype parameter values was not statistically different. APBPM assigned individual parameter values for each genotype, because the aim of its development was to allow it to be parameterised against arbitrary datasets.

BPM fitted a segmented regression model which allowed it to calculate the leaf expansion rate (LER) parameter which described the relationship between degree days and LAI. This step could not be reproduced in APBPM. To successfully calculate LER, the late season LAI measurements had to be removed by hand from the training dataset. This was not possible in an automated scenario, so LER was calculated by fitting a sigmoid curve to the data.

Both the parameterisation procedure and the model itself were implemented using Python 2.7 (Python Software Foundation, 2013), and R (R Core Team, 2017).

3.2.1 Model structure

Both BPM and APBPM were driven by meteorological data and used a set of differential equations to simulate changes in several variables occurring each day. A diagram of the models is shown in Figure 3.1.

The variables that were incremented daily are:

- Degree days ($°C\ day^{-1}$) – the cumulative degree days since plant emergence
- Leaf area index (dimensionless) – the total leaf area of the plant, divided by the area occupied by the plant:

$$\frac{\text{plant leaf area } (\text{m}^2)}{\text{area of ground occupied by plant}(\text{m}^2)}$$

- PAR (MJ m^{-2}) - photosynthetically active radiation, intercepted by the plant
- Dry matter yield (gm^{-2}) – grams of above ground dried biomass per m^2 of ground.

There were three genotype specific parameters that were used in this model: leaf expansion rate (LER), canopy extinction coefficient (k) and radiation use efficiency (RUE).

- LER (LAI °C day^{-1}) - describes the rate at which the leaf area index grows with respect to degree days.
- Canopy extinction coefficient (k) is a parameter that describes light interception
- RUE ($\text{g(dry weight) MJ}^{-1}\text{(PAR)}$) is the increase in dry weight (grams) for every MJ of intercepted PAR.

Parameterising the model meant finding values for the parameters. The model used the daily meteorological data, the parameter values and the equations to simulate the daily increment (delta) values for each variable:

- Degree days were calculated using the standard McVicker formula (McVicker, 1946), which takes the maximum and minimum daily temperatures and the base temperature (T_b). Similarly to (Hastings *et al.*, 2009), T_b was assumed to be 0°C for all genotypes, as that was the value used by (Davey *et al.*, 2015).

$$\Delta DD = f(T_{min}, T_{max}, T_b)$$

- LAI was simulated using the leaf expansion rate for the genotype and the daily degree days increment.

$$\Delta LAI = LER \times \Delta DD$$

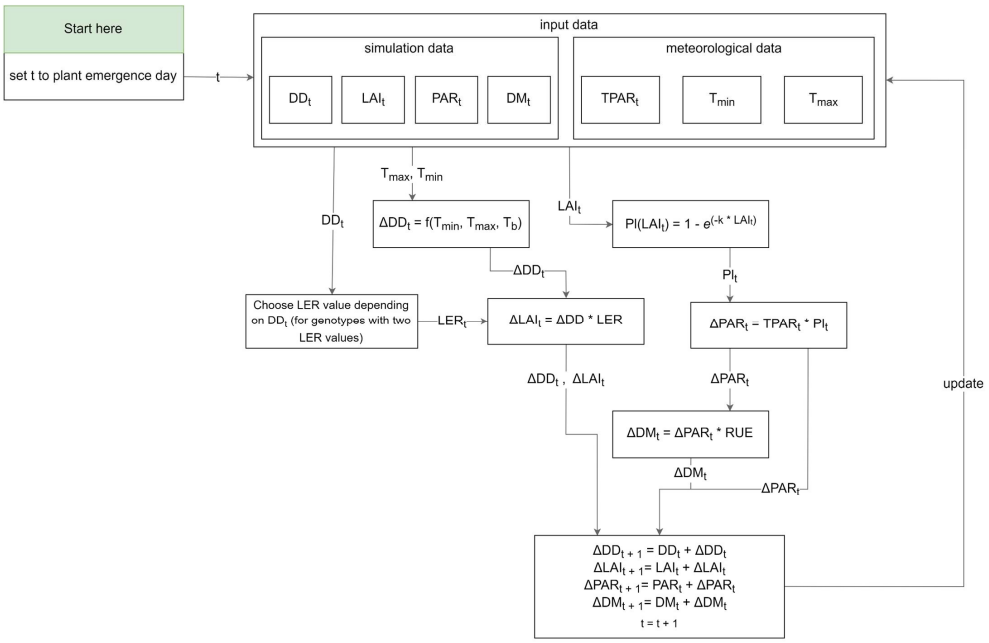

Figure 3.1 BPM model diagram. The four simulated variables (DD_t, LAI_t, PAR_t, DM_t) were set to 0 on emergence day. The model was stepped through daily and was driven by daily meteorological observations ($TPAR_t$, T_{min}, T_{max}). On each simulation day the coefficients k, RUE and LER were used to calculate the increment of the simulated variables. These were then updated and passed as input to the following day simulation

- The proportion of light (Pl) the plant had intercepted in that day was calculated using a function of the canopy extinction k of the genotype, and the LAI of the plant at the start of the day:

$$Pl = 1 - e^{(-k \times LAI)}$$

- Pl was used to simulate the PAR intercepted by the plant, which was a proportion of the total PAR on the day (TPAR) (taken from meteorological data)

$$\Delta PAR = TPAR \times Pl$$

- Finally, the increment of dry matter yield was expressed as a function of the intercepted $\Delta\mathrm{PAR}$ and the RUE for the genotype.

$$\Delta dry\ matter\ yield = \Delta PAR \times RUE$$

3.2.2 Model parameterisation

The following section features notes on the procedure used by (Davey *et al.*, 2015) for parameterising BPM. They are presented here to compare between the method used in the original publication and the method used in this project to parameterise APBPM. There are important differences between the modelling objectives in the original publication and this thesis. The aims behind parameterising BPM were to calculate and compare parameter values between genotypes and to establish whether they were significantly different. This dictated an approach where if genotype parameters were not statistically different from each other, the training data for the two genotypes would be combined and will be used to calculate a new common parameter value. The goal behind developing the automated training procedure was to enable parameterisation of BPM on arbitrary datasets. As comparison between the genotypes was not required, it was decided to take a different approach to the publication and assume that each genotype had a unique parameter value. That and other differences between the two models are described in detail in the following text.

LER

LER was initially estimated using the method described by (Davey *et al.*, 2015). This involved fitting the cumulative degree days since plant emergence (DD) to the LAI using either linear regression with the R lm() function or segmented regression from R "Segmented" package (Muggeo, 2008). The paper's authors suggested that this approach captured the two growth patterns that occurred in the genotypes. The first pattern, could be modelled by a single straight line, while the second was characterised by slow LAI growth phase earlier in the season, followed by a faster growth phase later in the season. Segmented regression fitted two lines to the data with a breakpoint of where one growth phase ended and the other started. If the first pattern occurred the LER value was taken to be the slope parameter of the linear regression. With the second pattern, the LER value was the slope of the first line if $\text{degree days} < \text{breakpoint}$, or the slope of the second line otherwise. Figure 3.2 demonstrates segmented regression applied to data from a single plot in the ABR61 trial, for year 2011.

During this study, it was discovered that later in the season genotypes exhibited a third growth phase, when growth stopped and LAI plateaued. Segmented regression as applied by (Davey *et al.*, 2015) failed to predict that stage resulting in overestimation of the LAI in late season. For the parameterisation done in the publication, the LAI measurements in the plateau phase were removed manually, to ensure a good fit for the second line of the segmented regression model. To replicate this approach with APBPM, an automatic way of choosing plateau data points to drop needed to be developed. This was not possible0 as the choice of these points was subjective.

Additionally, segmented regression is very sensitive to the number of data points, and often produced an error if certain data points were missing. As the aim of automating the parameterisation was to make it possible to train the model with a reduced set of data points, it was infeasible to use segmented regression. LER was instead modelled using a sigmoid function (Verhurlst, 1838), defined by the formula:

$$LAI = \frac{L}{(1+e^{\left(-q \times (degree\ days - x_{mid})\right)})}$$

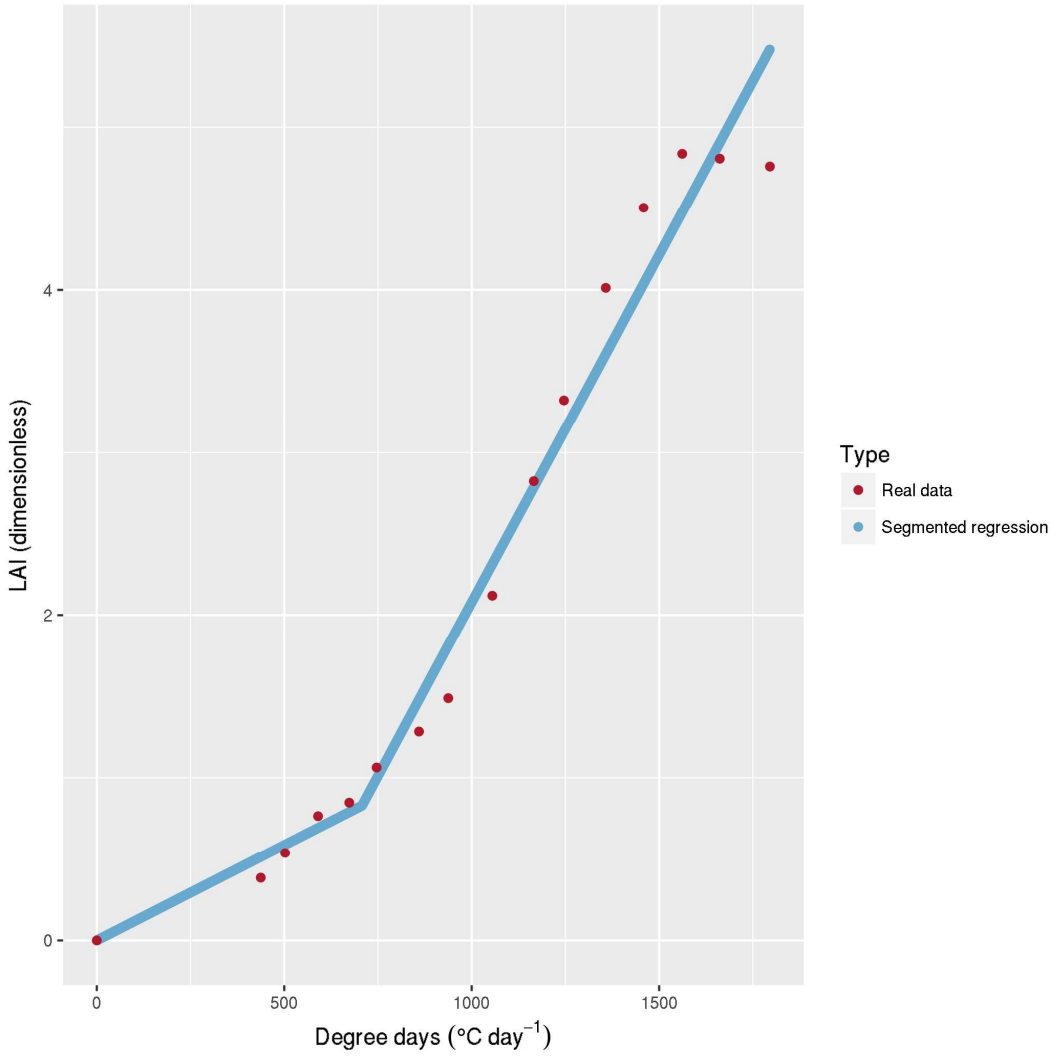

Figure 3.2 Applying segmented regression for fitting degree days to LAI. The data is a single plot of a M. sacchariflorus genotype (Sac-5), from the ABR61 trial, year 2011. The change between the two growing phases as modelled by segmented regression, occurs at its breakpoint at degree days = 121.7. LER is estimated to be the slope of the first line when degree days < 121.7 and the slope of the second line otherwise.

L. q and $\mathrm{x_{mid}}$ were parameters of the sigmoid function, that were fit using non-linear least squares regression, using the R function "nlsLM" from the R library "minpack.lm" (Elzhov *et al.*, 2016). Figure 3.3 (top panel) illustrates an example sigmoid function, with parameters: L = 5, q = 0.02, $\mathrm{x_{mid}}$ = 250. L controls the maximum value reached by the function, q controls function steepness, and $\mathrm{x_{mid}}$ is the point where the function has the highest slope.

The function was chosen for its ability to capture the initially slow growth, which increases in the middle and decreases and stops towards the end of the growing season.

LER was estimated as the slope of the function at a given point, which was calculated using the first derivative of the sigmoid function. The first derivative was defined as:

$$LER(degree\ days) = \frac{L \times q \times e^{-q \times (degree\ days - x_{mid})}}{(1 + e^{-q \times (degree\ days - x_{mid})})^2}$$

The given point was the degree days value passed to the function. To calculate LER for each day in the simulation, it was taken as the cumulative degree days from plant emergence until the beginning of that day. This means that the value of LER changed depending on the time of the season. The value of LER and its change against degree days is illustrated for the example curve on the bottom panel of Figure 3.3.

LER in BPM was estimated using cumulative degree days and LAI measurements from year 2011, whereas APBPM used data from both 2011 and 2012. This was done to ensure a more objective procedure – BPM did use data from 2012, but only to parameterise k as there was not enough data in 2011 due to issues with the SunScan instrument. Instead of programming in special cases for using some of the data for some parameters only, it was decided to provide APBPM with all available data for training. The parameterisation procedure of the sigmoid function is summarised in Figure 3.4. The input data was mean LAI and degree day measurements per plot from years 2011 and 2012. The output L, q, and $\mathrm{x_{mid}}$ parameters of the fitting procedure were used for estimating LER during simulation using the $\mathrm{LER(degree\ days)}$ function described above.

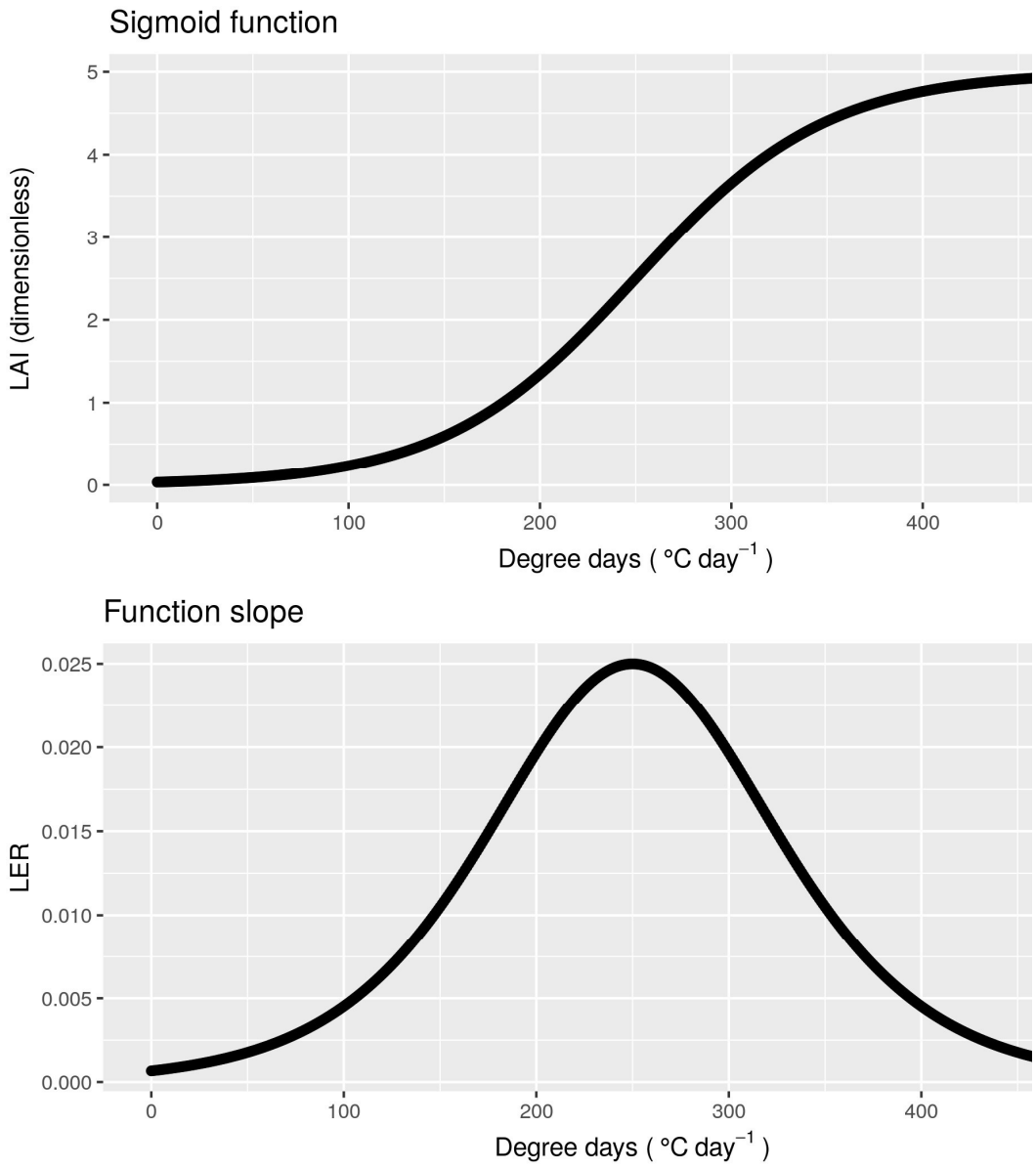

Figure 3.3 Example LAI and LER simulation using a sigmoid function with parameters: L = 5, q = 0.02, x_{mid} = 250.

Top panel: LAI simulated by the sigmoid function

Bottom panel: LER values for corresponding degree days

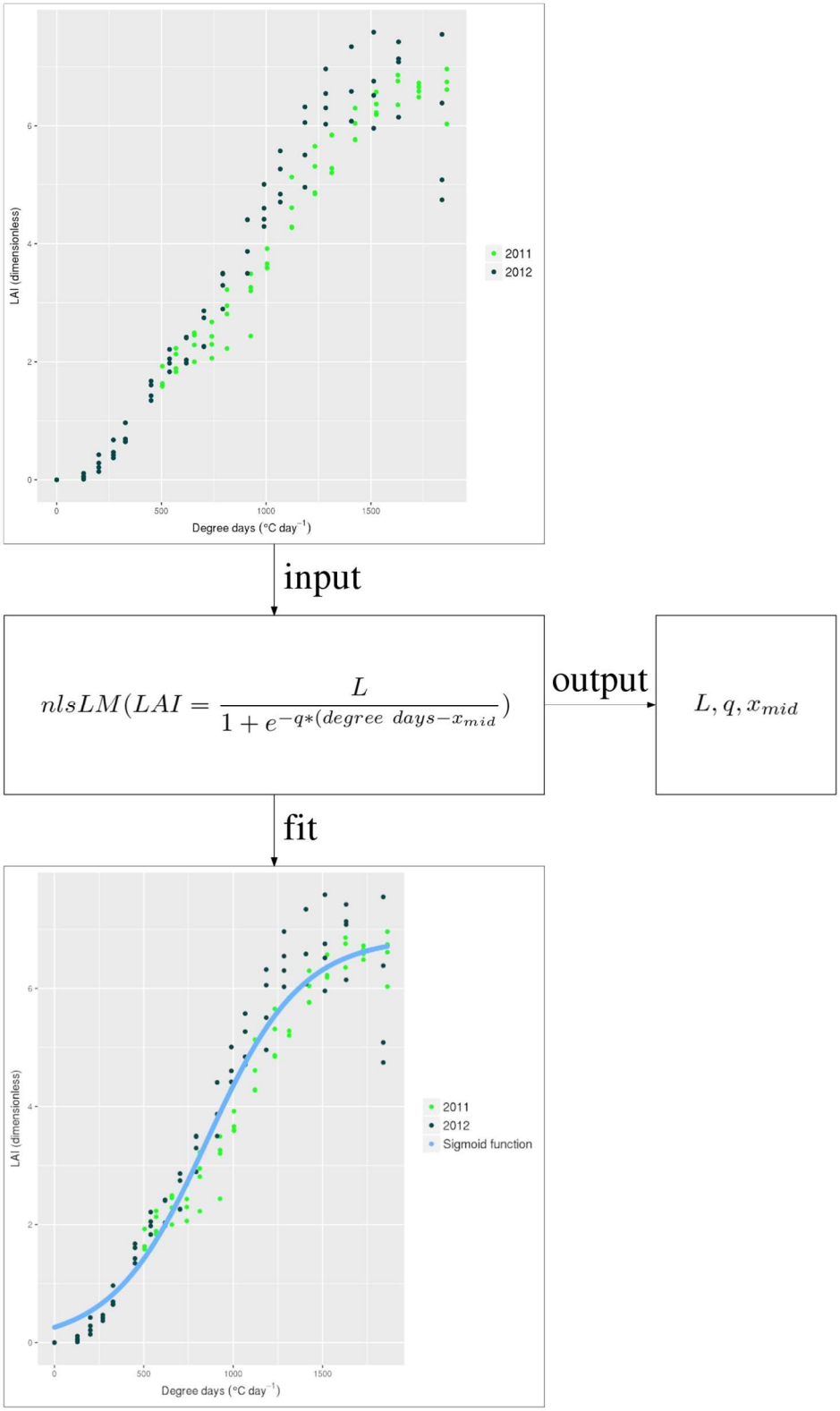

Figure 3.4 Parameterisation procedure for the sigmoid function for each genotype. The input data was degree day and LAI measurements from years 2011 and 2012. The data used in this example is of all plots of M. x. Giganteus, years 2011 and 2012. The output L, q and x_{mid} parameter values were used in simulations for estimating the LER value for the corresponding genotype.

The procedure described above differs from the approach used in the research paper that described BPM (Davey *et al.*, 2015). There BPM was parameterised, by applying linear and segmented regression to each individual plot. As there were four replicates, four LER values were calculated for each genotype. The paper authors compared LER values were between genotypes to see if they were statistically different using ANOVA. For genotypes where that was not the case, mean LER across these genotypes was assigned as their final LER value.

While this approach was required for BPM, to compare genotypes and their parameter values, this was not necessary for the automation procedure. For this reason, when training APBPM the plot data from both years was pooled together and the sigmoid function was fitted to it. This also simplified the process and reduced the computation time.

k

Real measurements of proportion of light were provided by the transmission measurements (SunScan data). Transmission and LAI were used to determine k, by fitting a curve to the data using the transmission equation (Monsi and Saeki, 1953) below and the "nls" function from the "stats" package (R Core Team, 2017).

$$Transmission = e^{(-k \times LAI)}$$

As transmission is the proportion of light not intercepted by the plant, the proportion of light (PI) that was intercepted is $(1 - \text{Transmission})$. Substituting transmission in this equation gives the PI equation in the model. The parameterisation procedure is illustrated on Figure 3.5. The input data was mean LAI and transmission values for each plot from years 2011 and 2012. The fitting procedure outputted the value for the k parameter for each genotype.

In BPM, pairs of genotypes were compared to determine if they had a common k value, by first fitting "nls" to each genotype individually and then to both genotypes together. An F-test was used to determine if the k values were significantly different. With APBPM, it was assumed that each genotype had a unique k value, which simplified the process to just fitting "nls" individually. The outlier points on Figure 3.5

suggest that there could be a difference in k values between years, however testing this was outside of the score of this thesis.

RUE

As mentioned before, RUE described the efficiency of the plant to use the intercepted PAR to increase its biomass, as defined in the equation below:

$$dry\ matter\ yield = PAR \times RUE$$

To find the right value for this parameter, values for dry matter yield and corresponding cumulative intercepted PAR were needed. Dry matter yield data was collected through harvesting, but it was impossible to directly measure cumulative intercepted PAR. However, the values for LER and k were already calculated and this made it possible to use a reduced version of the model to simulate cumulative intercepted PAR by the plant.

The reduced version of the model simulated all variables except yield and was run using a daily time step. Figure 3.6 illustrates the model structure. The simulated intercepted PAR was fitted to real dry matter yield with linear regression, using the "lm" function in R and the equation:

$$dry\ matter\ yield = PAR \times RUE + 0$$

The intercept was forced to 0 as when the cumulative PAR was 0 the yield was also assumed to be 0, as the plant had not intercepted any light yet. The slope term of the model was the value for RUE. The parameterisation procedure is shown in Figure 3.7.

It was assumed that each genotype had a unique RUE value. This is in contrast with the approach taken for parameterising BPM, where both linear regression and ANCOVA were used to fit an individual value for each genotype or data from some genotypes was pooled and a common RUE value was assigned to them. The method for BPM resembled the one used by (Davey *et al.*, 2015) for finding LER values, described above.

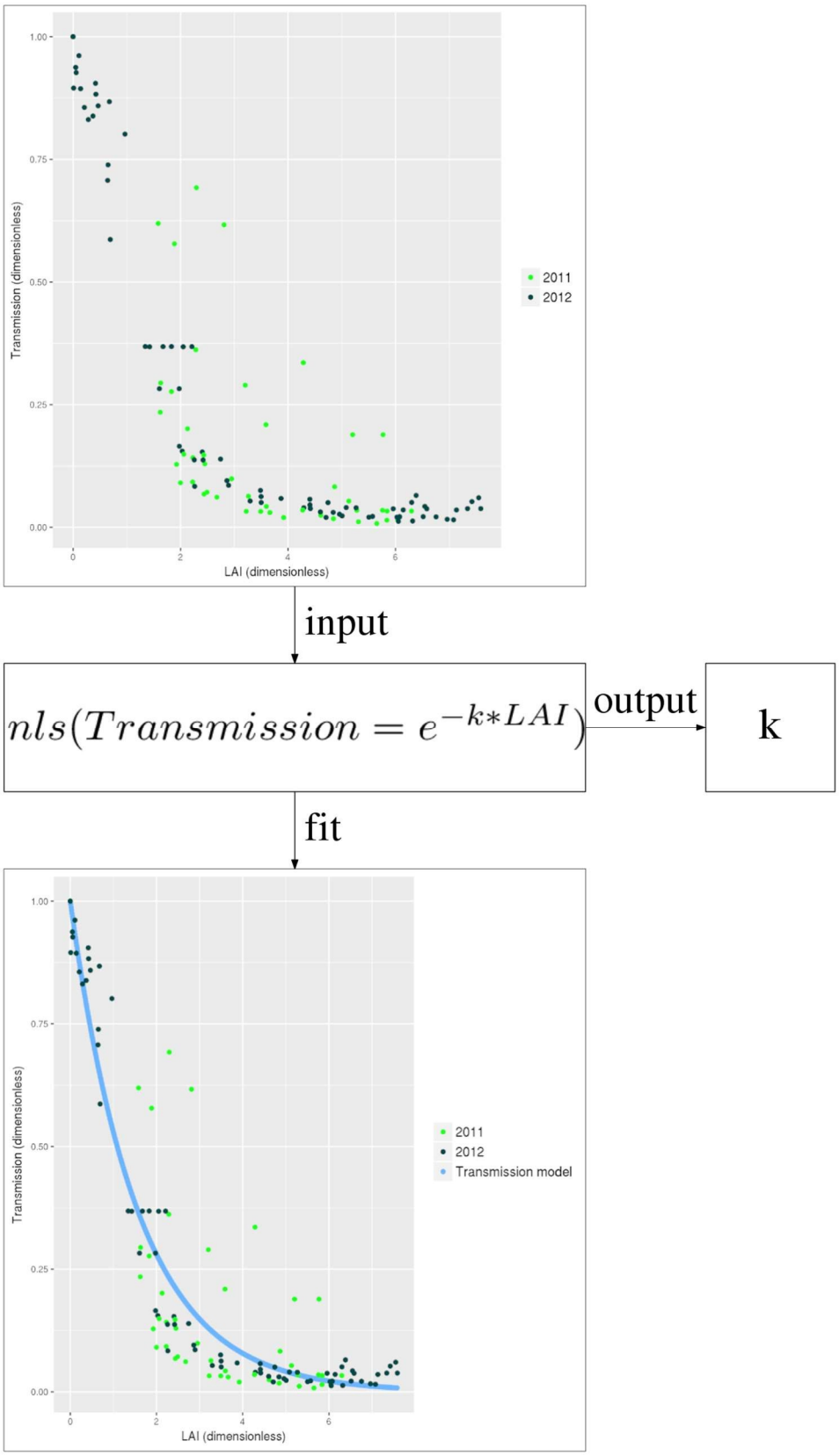

Figure 3.5 Parameterisation procedure for finding the k value for each genotype. The input data was transmission and LAI measurements from years 2011 and 2012. The data used in this example is of all plots of M. x. Giganteus, years 2011 and 2012. The output k parameter value was used in simulations for estimating the proportion of intercepted light for the corresponding genotype.

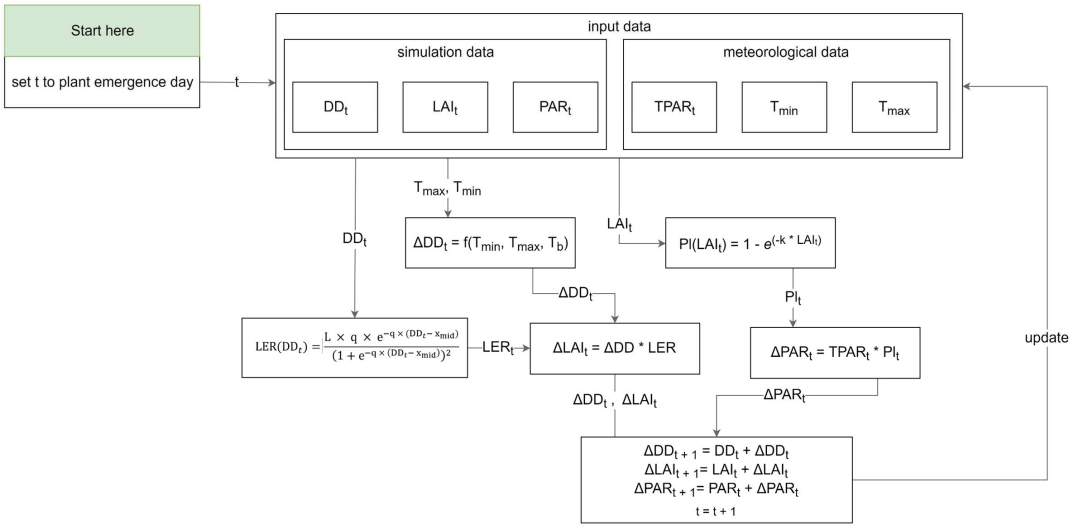

Figure 3.6 Reduced model version. Used for simulating values for PAR that are then used for calculating the RUE values. The reduced model was stepped through daily, using the real meteorological observations ($TPAR_t$, T_{max}, T_{min}) from that day (t) and the simulated values (DD_t, LAI_t, PAR_t) at the start of the day to simulate the daily increments (ΔDD, ΔLAI, ΔPAR) used to update the simulated variables. LER is calculated using the first derivative of the sigmoid function, given the degree days at the beginning of the day.

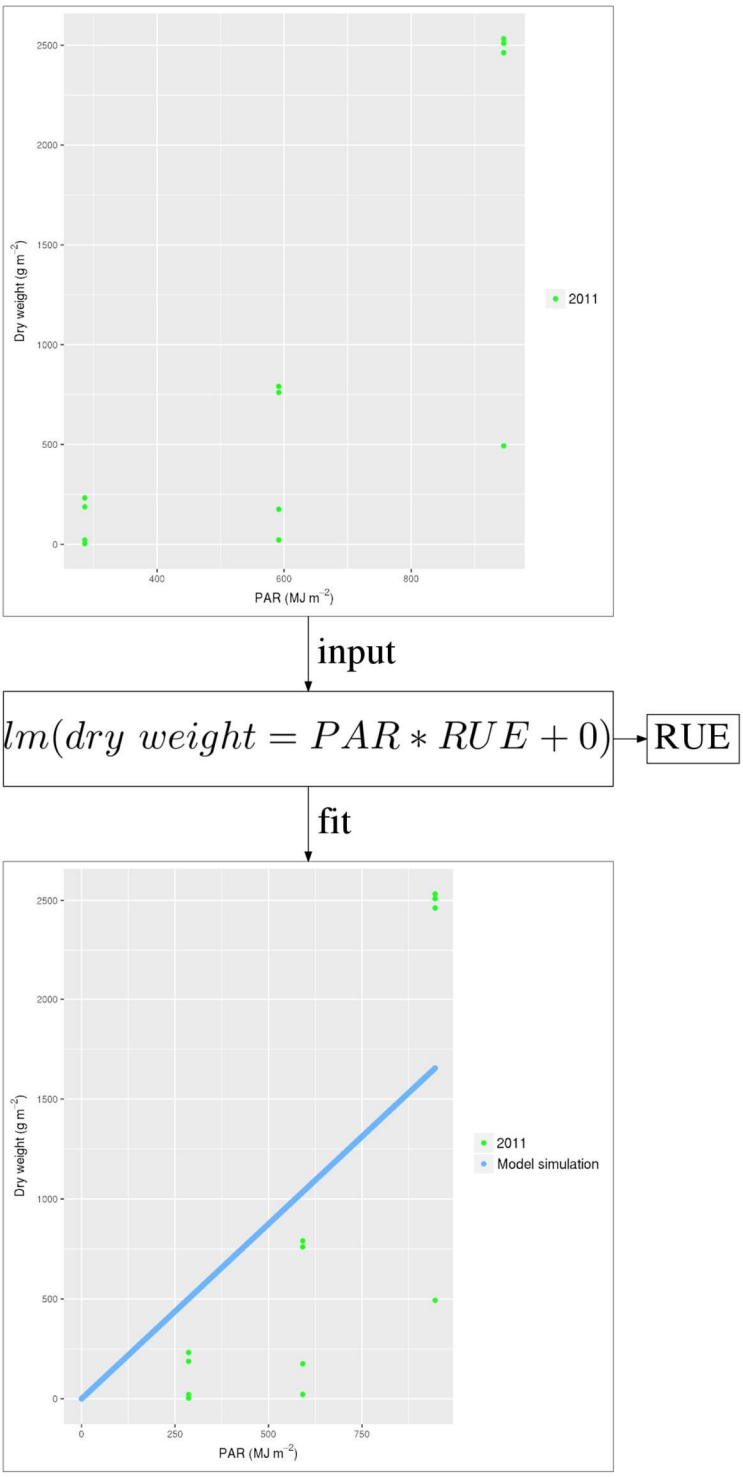

Figure 3.7 Parameterisation procedure for RUE for each genotype. The input data was simulated intercepted PAR and real dry weight measurements from 2011 (as no dry weight measurements were done in the 2012 growing season). A simple linear regression with intercept forced through 0 was fitted using the formula in the graph. The data used in this example is of all plots of M. x. Giganteus. The output RUE parameter was used in APBPM simulations for predicting dry matter yield. As can be seen from the plotted data, the relationship between simulated cumulative PAR and dry weight was not linear, so the fitted model did not capture that relationship very well. In the future, the RUE parameter could be validated and the relationship between PAR and dry weight could be examined in more detail.

3.2.3 Simulations and validation

As mentioned before, the data collected during the years 2011 and 2012 from the ABR61 trial was provided as a training dataset for APBPM. This was the same dataset used by (Davey *et al.*, 2015) for parameterising BPM. To establish how similar the two models are, the values for the genotype specific coefficients were compared across the two models, where that was possible.

Simulations using BPM and APBPM were compared, to further validate the automated procedure as an accurate reproduction of the parameterisation from the publication. To perform simulations using BPM, it was implemented in Python using the parameters from the publication. Simulations were done on each year within the ABR61 dataset (2011, 2012, 2014, 2015 and 2016). The final structure of APBPM used for these simulations is illustrated in Figure 3.8. The approaches used for parameterising APBPM and performing simulations using the two models are summarised in Figure 3.9 and Figure 3.10. For each simulated variable in the models (LAI, proportion of light, PAR intercepted by the plant and dry matter yield), the coefficient of determination (R^2) and RMSE were calculated for simulated values by APBPM against simulated values by BPM. This provided information on how similar the two models were in their predictions. The two statistics were calculated both across all genotypes and for each specific genotype, to determine any genotypic effects. Degree days were not included in the comparison as the models used the same formula and data to calculate them, which meant that the values across the models were the same.

The two models were compared in their prediction accuracy for the simulated variables. Their values were compared with real data, using R^2 and RMSE statistics. They were calculated separately over the training data (years 2011 and 2012) and test data (years 2014, 2015 and 2016) to determine the loss of accuracy when predicting unseen data.

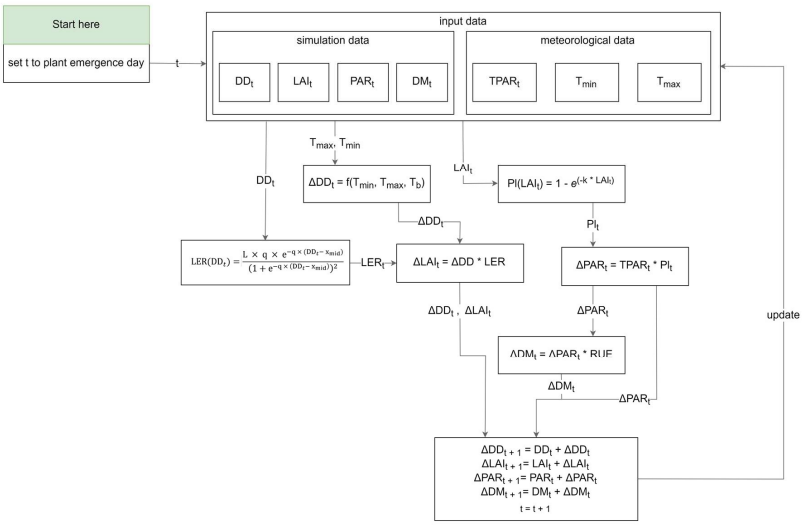

Figure 3.8 Model structure of the APBPM model. The four simulated variables (DD_t, LAI_t, PAR_t, DM_t) were set to 0 on emergence day. The model was stepped through daily and was driven by daily meteorological observations ($TPAR_t$, T_{min}, T_{max}). On each simulation day the genotype specific coefficients (k, RUE and the sigmoid parameters L, q and x_{mid} were used to calculate the increment of the simulated variables. These were then updated and passed as input to the following day simulation.

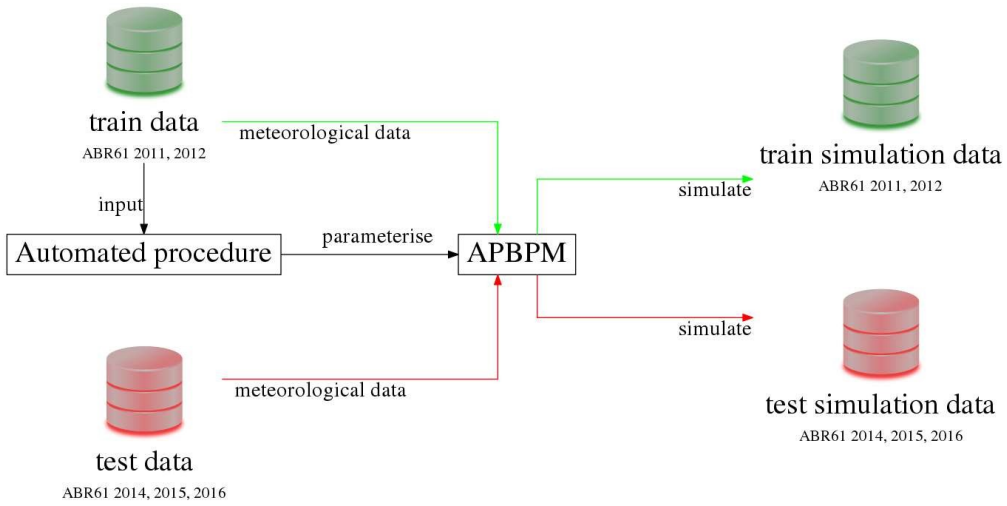

Figure 3.9 Training and simulation procedure for APBPM. The training dataset (years 2011 and 2012) is provided to the automated procedure which parameterises APBPM. Meteorological data from the training dataset is provided directly to APBPM and used to drive the simulations of years 2011 and 2012. The meteorological data from the test dataset is given to APBPM to simulate the years 2014, 2015 and 2016.

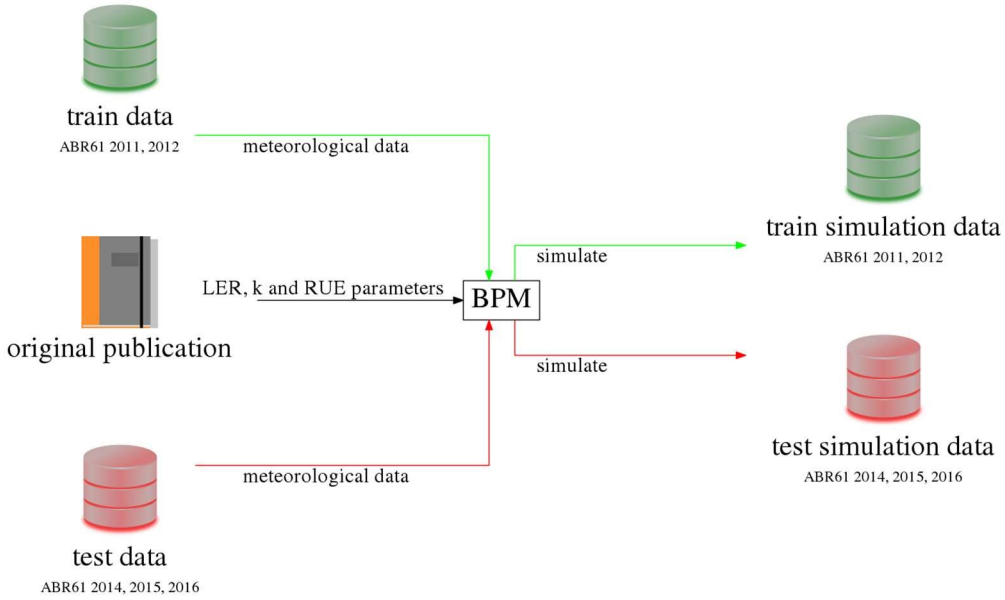

Figure 3.10 Simulation procedure for BPM. Genotype specific parameters needed for the model are taken directly from the publication. Meteorological data from the training dataset is provided directly to the model and used to drive the simulations of years 2011 and 2012. The meteorological data from the test dataset is given to BPM to simulate the years 2014, 2015 and 2016.

3.3 Results

3.3.1 LER and LAI

Due to the differences of LER estimation between BPM and APBPM a direct comparison of values could not be made. However, it was possible to compare the simulated LAI values from the two models. Figures 3.11 and 3.12 demonstrate the two models' LAI simulation performance over the training dataset (years 2011 and 2012 respectively). The coefficient of determination R^2 between their predictions was 0.859 and the RMSE was 1.616. Coefficients of determinations and RMSE for each individual genotype are presented in Table 3.1.

The two models mostly agreed with each other, but late in the growing season BPM predicted a continued growth, whereas APBPM's LAI reached a plateau and was closer to the real values. This was particularly evident in Goliath and Sac-5 on Figure 3.11 and explained their lower R^2 and higher RMSE. A comparison between all simulated LAI values generated by the two models demonstrated this further (Figure 3.13). The significant differences in the high LAI ranges were due to the plateau growth phase being modelled by APBPM. The smaller differences in the lower LAI ranges (values below 1) were caused by the shape of the sigmoid function, which resulted in estimating early season growth to be slower than what BPM provided.

As mentioned by (Davey *et al.*, 2015) EMI-11 behaved differently in 2011 and 2012. This made it difficult for the models to find an LER value that worked for both years and resulted in both models underestimating the true LAI in 2012.

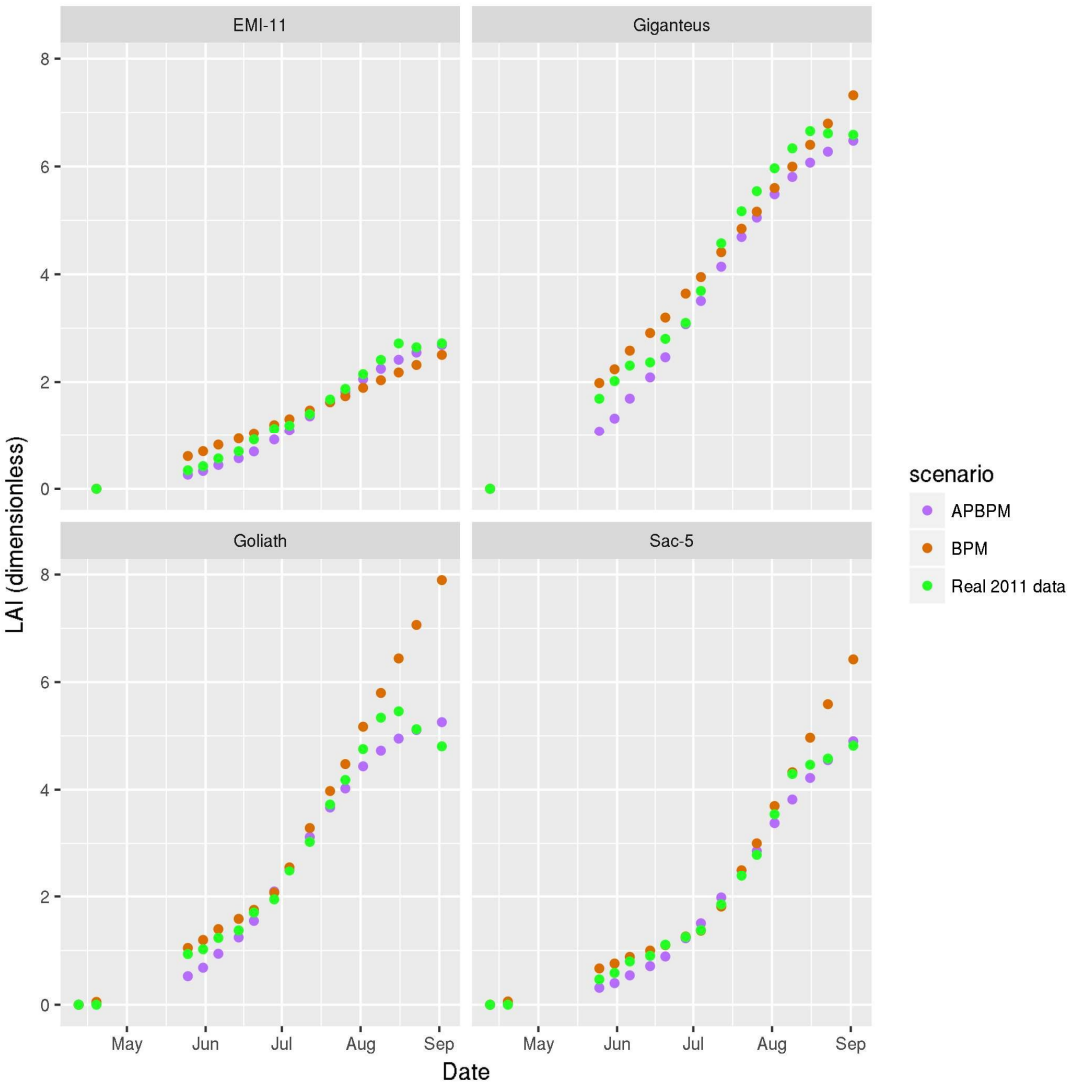

Figure 3.11 Simulated and actual mean cumulative LAI values per genotype for the year 2011.

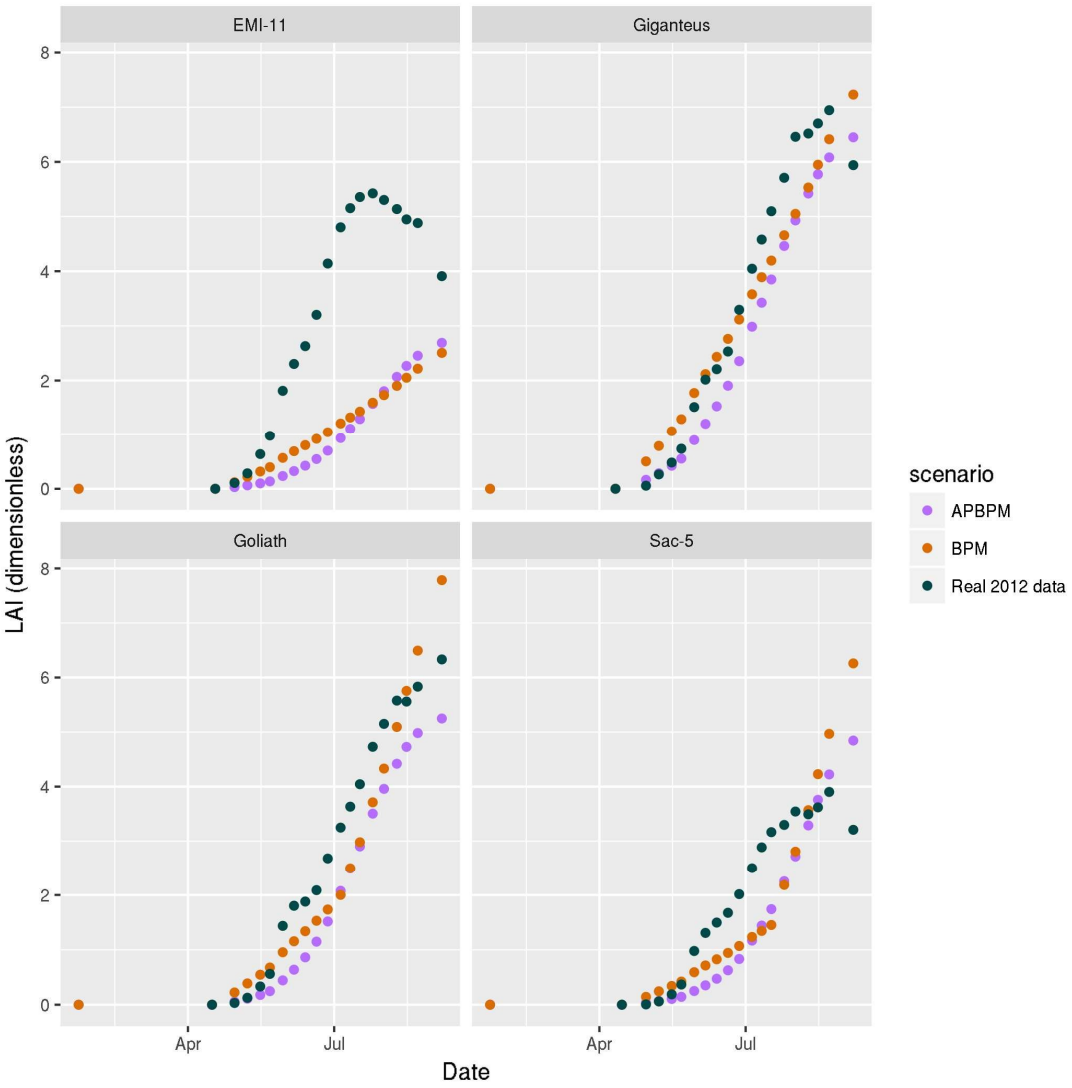

Figure 3.12 Simulated and actual mean LAI values per genotype for the year 2012.

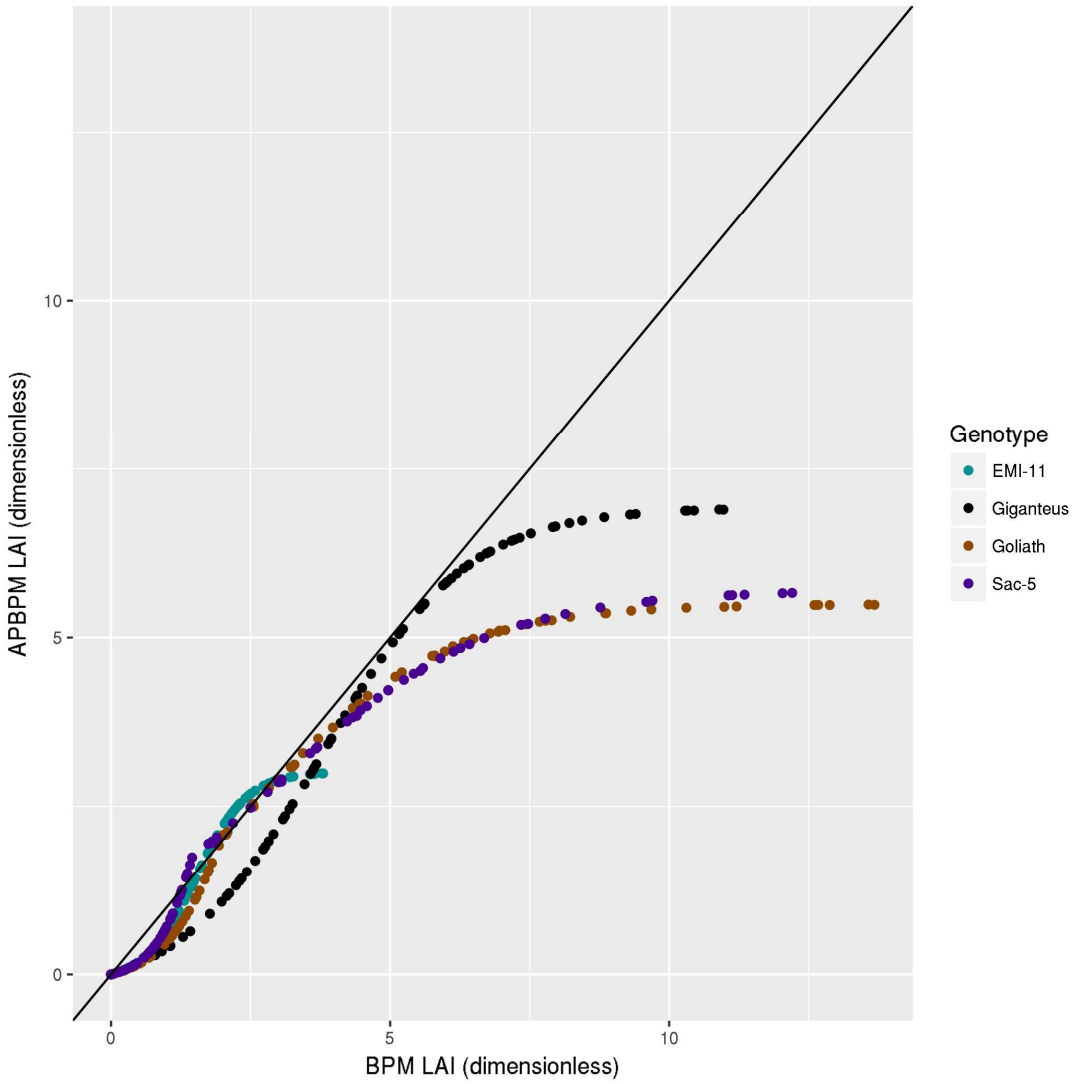

Figure 3.13 LAI values generated by APBPM vs those generated by BPM from all simulations (both on the training and the test dataset). The plateau phase modelled by APBPM caused divergence from the values that BPM gave. There is a clear genotyping effect as the plateau happened at different times of the season. The overall (over all genotypes pooled together) coefficient of determination R^2 between the model predictions was 0.859 and the RMSE was 1.616. Table 3.1 shows the individual R^2 and RMSE for each genotype. The difference between the LAI values from the two models was caused mainly by the use of the sigmoid function in APBPM to simulate LAI.

	Giganteus	Goliath	Sac-5	EMI-11
R^2	0.902	0.814	0.884	0.936
RMSE	1.07	2.15	1.54	0.28

Table 3.1 Coefficients of determination R^2 and RMSEs between the LAI simulated values from the two models for each individual genotype.

While the predictions of APBPM were different from those of BPM, the procedure used for APBPM improved modelling precision, especially on unseen test data (years 2014, 2015 and 2016). The results are summarised in Table 3.4. To put the RMSE values in perspective Table 3.2 and Table 3.3 show descriptive statistics for LAI in the training and test dataset respectively. A scatter plot of simulated LAI vs real LAI values of both datasets together is shown in Figure 3.14. The real LAI values in the figure were the average LAI values for each plot. As the model simulated on a genotype level, a single simulated LAI value for a given date, corresponded to four real plot average LAI values, for the four different plots containing that genotype.

While the performance over the training dataset was roughly the same. the improvement was more dramatic over the test dataset. There were multiple reasons for the overall increase in modelling error in the test data. Some loss of modelling accuracy was expected when predicting unseen data – environmental and climatic conditions might be outside of the scope of the training dataset. But a crucial factor for accuracy is maturity of the plants. The trial was established in March 2009, so during 2011 and 2012 they were 2 and 3 years old. It was suggested that some immaturity effects were present (Davey *et al.*, 2015), which may have caused the variation in EMI-11 growth between 2011 and 2012. Training on potentially immature plants and predicting their growth when they are mature (test dataset) is likely to have reduced accuracy further. An example is illustrated with predictions for year 2014 shown in Figure 3.15, where the growth patterns of the genotypes were significantly different to what the models were estimating. It is evident that the sigmoid function used to model the relationship between DD and LAI did not reflect the truth – as the plants senesced at the end of the growing season LAI declined. However, changing the model further to allow it to predict the decline in LAI was outside of the remit of the work described in this chapter – the objective of the automated procedure was to produce a model equivalent to BPM, to prove that this process can be automated. Any further improvements would have produced a significantly different model.

Genotype	Mean	Standard deviation	Minimum value	25%	50%	75%	Maximum value
EMI-11	2.54	1.89	0.04	0.92	2.15	3.59	6.82
Giganteus	3.93	2.21	0.01	2.1	3.92	6.05	7.59
Goliath	3.08	2.15	0	1.21	2.88	4.5	8.56
Sac-5	2.19	1.69	0	0.86	1.91	3.14	7.16

Table 3.2 Descriptive statistics for the LAI values in the training dataset (years 2011 and 2012). The percentile columns denote the first quartile (25%), the second quartile (median, 50%) and the third quartile (75%).

Genotype	Mean	Standard deviation	Minimum value	25%	50%	75%	Maximum value
EMI-11	4.54	2.5	0.03	2.68	4.95	6.25	11.43
Giganteus	4.76	3.53	0	1.71	4.56	7.23	13.35
Goliath	4.52	3.78	0.01	1.17	3.8	7.26	13.67
Sac-5	3.59	1.95	0.01	2.23	3.89	4.93	8.39

Table 3.3 Descriptive statistics for the LAI values in the test dataset (years 2014, 2015 and 2016).

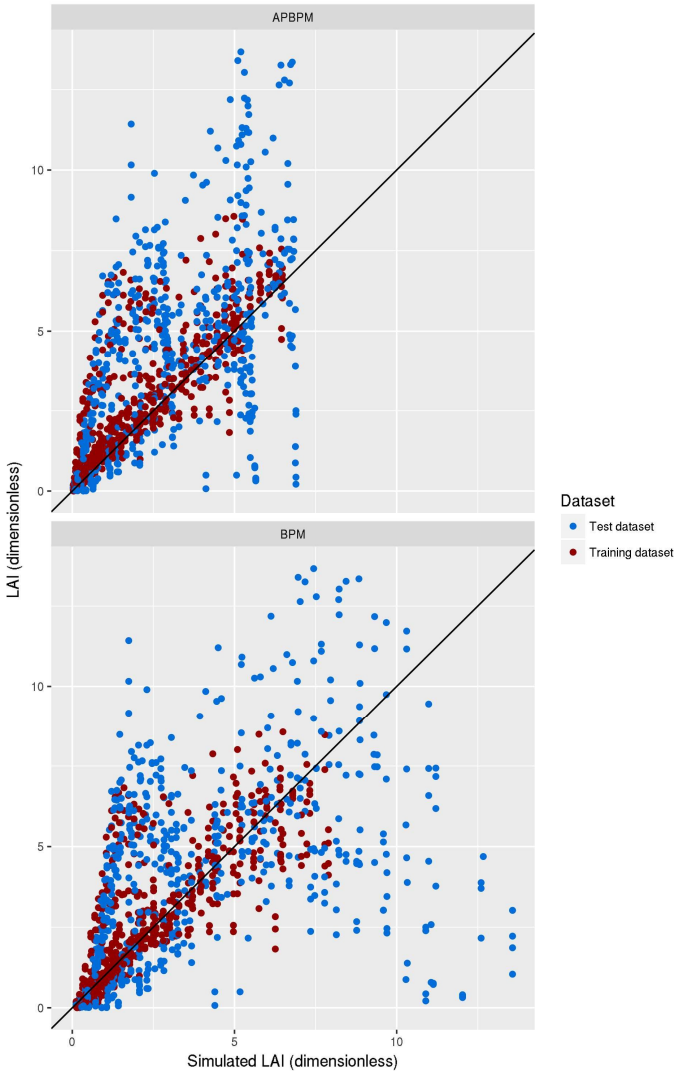

Figure 3.14 Comparison between simulated LAI values generated by the two models and actual LAI observations. The red points are predictions made on the training dataset compared to real observations from the training data, and the blue points are predictions on the test dataset compared to real observations from the test data. Table 3.4 lists the R^2 and RMSE for these models over both datasets.

Top panel: APBPM simulated vs actual LAI values

Bottom panel: BPM simulated vs actual LAI values

	Training dataset		Test dataset	
	R^2	RMSE	R^2	RMSE
BPM	0.600	1.41	0.172	3.42
APBPM	0.662	1.4	0.327	2.85

Table 3.4 LAI simulation accuracy results, for the training dataset (years 2011 and 2012) and the test dataset (years 2014-2016)

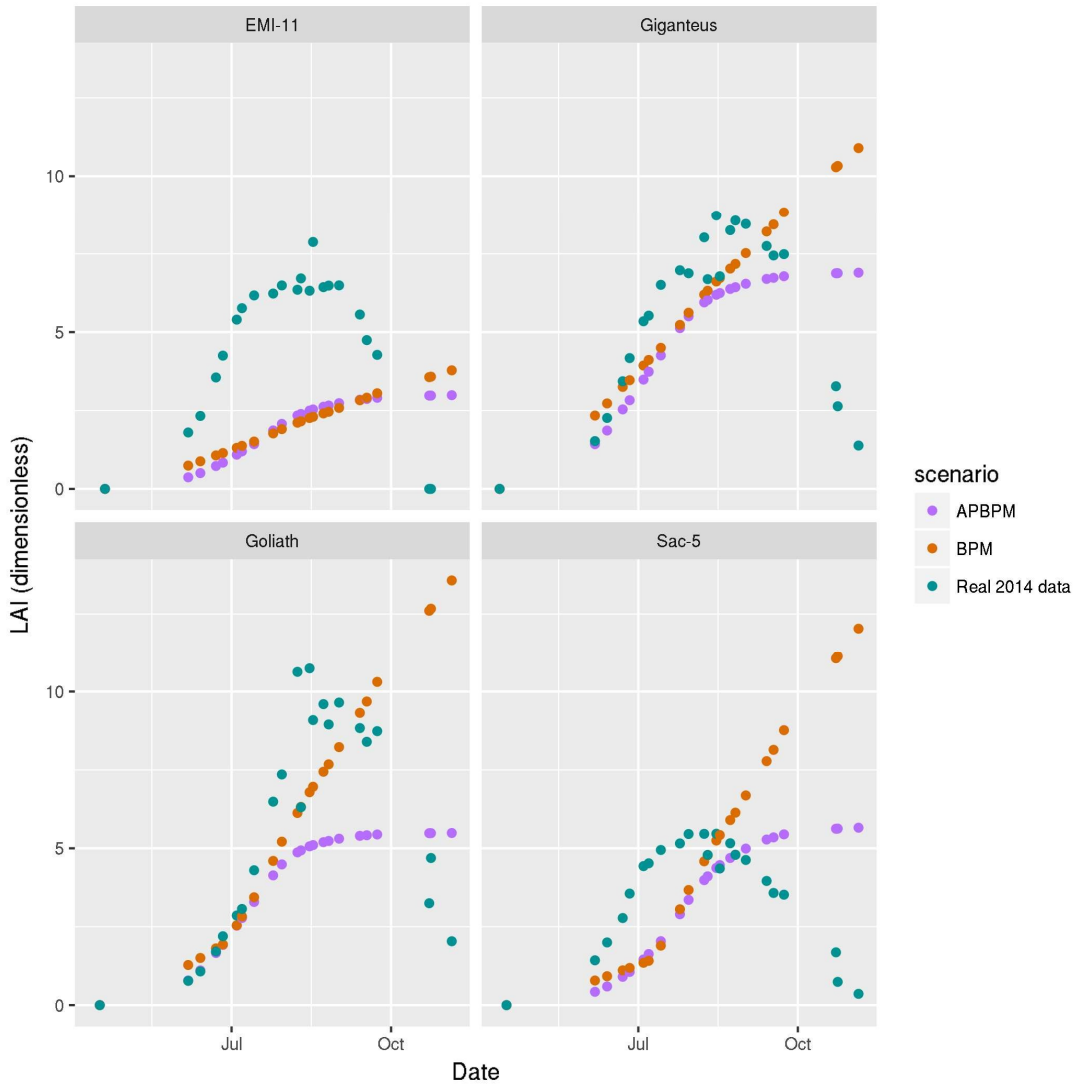

Figure 3.15 Simulated and actual mean LAI values per genotype for the year 2014. In many cases APBPM's simulated values plateau lower than the actual – this is likely due to the maturity effect, meaning that plants reached higher LAI in their mature years (2014 and later) compared to the training data (years 2011 and 2012).

Additionally, poor establishment may have affected the performance of the genotypes in 2011 and 2012, but by the later years (2014-2016) covered by the test data, that may have no longer been an issue. This, however, was not tested as it was outside of the scope of this project. Also, in the test dataset, measurements were continued in October, so the loss of LAI due to senescence was captured. Late season data was thus not modelled well by the two models, as none of them accounted for this growth pattern.

3.3.2 k and light interception

	Giganteus	Goliath	Sac-5	EMI-11
BPM 2011	0.6539	0.5533	0.6539	1.129
BPM 2012 and later	0.6539	0.5533	0.6539	0.5533
APBPM	0.6365	0.5677	0.6360	0.7670

Table 3.5 k coefficient values for the two models. BPM values were taken directly from (Davey *et al.*, 2015)*. The 2012 values for BPM were used for simulating later years as well.*

The k coefficients for the two models are listed in Table 3.5. BPM had two separate k values for EMI-11, one for each year of the training dataset. For the BPM simulations done in this project, the first value was used only for simulating the year 2011 and the second value was used for years 2012 and later. This was done because of the dramatic difference in growth between the two years for that genotype – the observed relationship between transmission and LAI for EMI-11 in the year 2011 was very different from 2012. This is shown on Figure 3.16. While a difference emergence could cause such a difference in growth, the emergence day for EMI-11 in 2011 and 2012 was the same. One plausible explanation for the observed difference is that it could have been due to plant immaturity – the authors of BPM noted that compared to other genotypes, EMI-11 matured slowly (Davey *et al.*, 2015).

For three of the genotypes (Giganteus, Goliath and Sac-5) the k values of BPM and APBPM were very close to each other. For EMI-11, APBPM's k value was close to the average between the two values from BPM. This was expected, as APBPM estimated its k value using data from 2011 and 2012 together. While the k values were close, the two models differed in their predictions of transmission, and subsequently PI. The simulated and real transmission values for 2011 and 2012 are presented in Figure 3.17 and Figure 3.18 respectively.

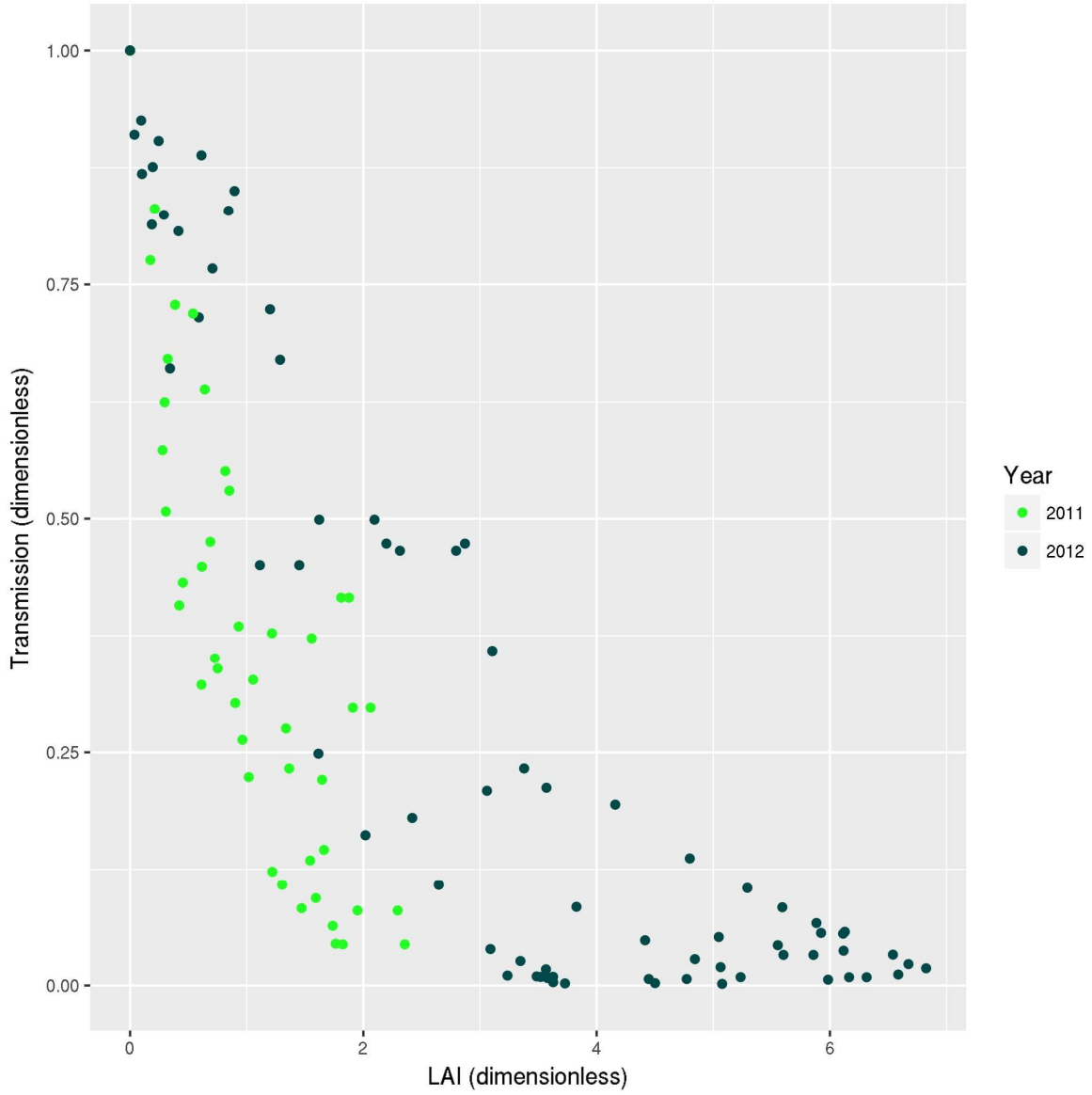

Figure 3.16 LAI vs transmission values for EMI-11 during 2011 and 2012. The difference between years in the relationship between these two variables prompted the authors of BPM (Davey *et al.*, 2015) *to fit two different k values for EMI-11, one for each year. Because APBPM fitted a single individual parameter value for each genotype, it did not simulate this relationship very well.*

The difference in canopy closure can be seen from a visual comparison of the real transmission values in Figure 3.18 – Giganteus and EMI-11 achieved almost complete canopy closure by the beginning of July, allowing them to intercept all available light. Goliath did that in late August, and Sac-5 never reached that point. BPM simulated transmission for EMI-11 in 2011 a lot better than APBPM – this was most likely due to BPM fitting two k parameters instead of just one for this genotype. The two models' predictions are compared in Figure 3.19. The coefficient of determination R^2 between the predictions of the two models was 0.899 and the RMSE was 0.109. The statistics for each genotype is listed in Table 3.6.

The lower correlation in EMI-11 was due to the use of a single k value for EMI-11 in APBPM, as opposed to two in BPM. The discrepancies in the other genotypes were mainly caused by the different approach to modelling LAI. As predictions of transmission depended both on k values and simulated LAI, any differences in LAI predictions between the two models would propagate through the model and cause differences in simulated transmission. However, while the two models' LAI values differed most during late season, due to the sigmoid plateau, this only had limited influence over transmission discrepancies. A look into the relationship between simulated LAI and transmission values (Figure 3.20) demonstrates why that is the case – for LAI values larger than 4, transmission values were low, they changed very little and the plant was already considered to be intercepting more than 90% of the light ($\mathrm{Pl} = 1 - \mathrm{transmission}$ as mentioned in section 3.2.2). Thus, the biggest discrepancies between the models were present in early season simulations where LAI values were small and transmission decreased quickly. This means that above a certain LAI threshold, any differences in LAI in the models would have had minimal effects over simulations of transmission, proportion of light Pl (as it is directly dependent on transmission), intercepted PAR and yield.

Despite the differences, APBPM still managed to improve the accuracy of transmission simulation. While, the scatter plots for the two models' simulated transmission against real values looked very similar to each other (Figure 3.21), the simulations by APBPM were more correlated in both the training and test dataset as illustrated in Table 3.9. Tables 3.7 and 3.8 show descriptive statistics for the transmission values in the training and test datasets respectively.

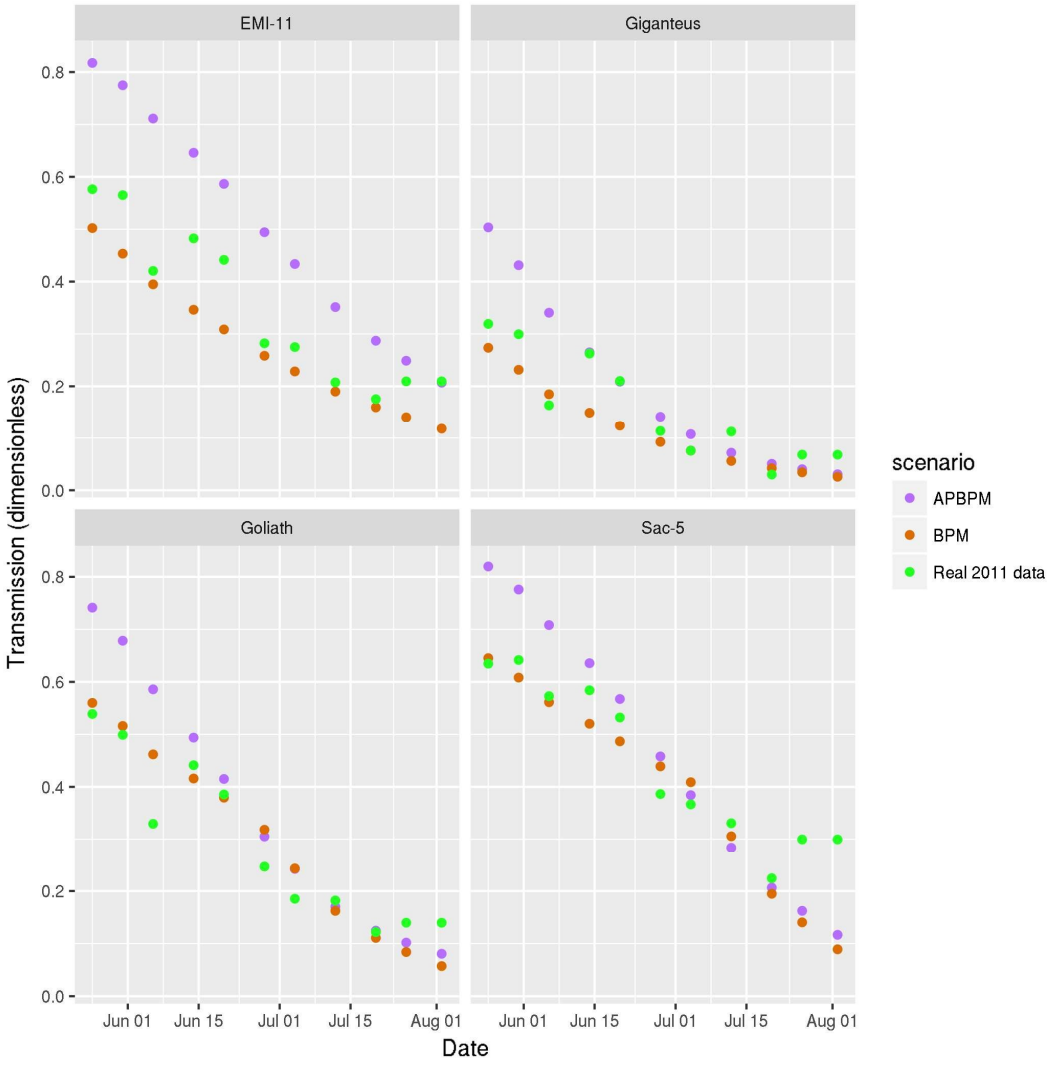

Figure 3.17 Simulated and actual mean transmission values per genotype for the year 2011

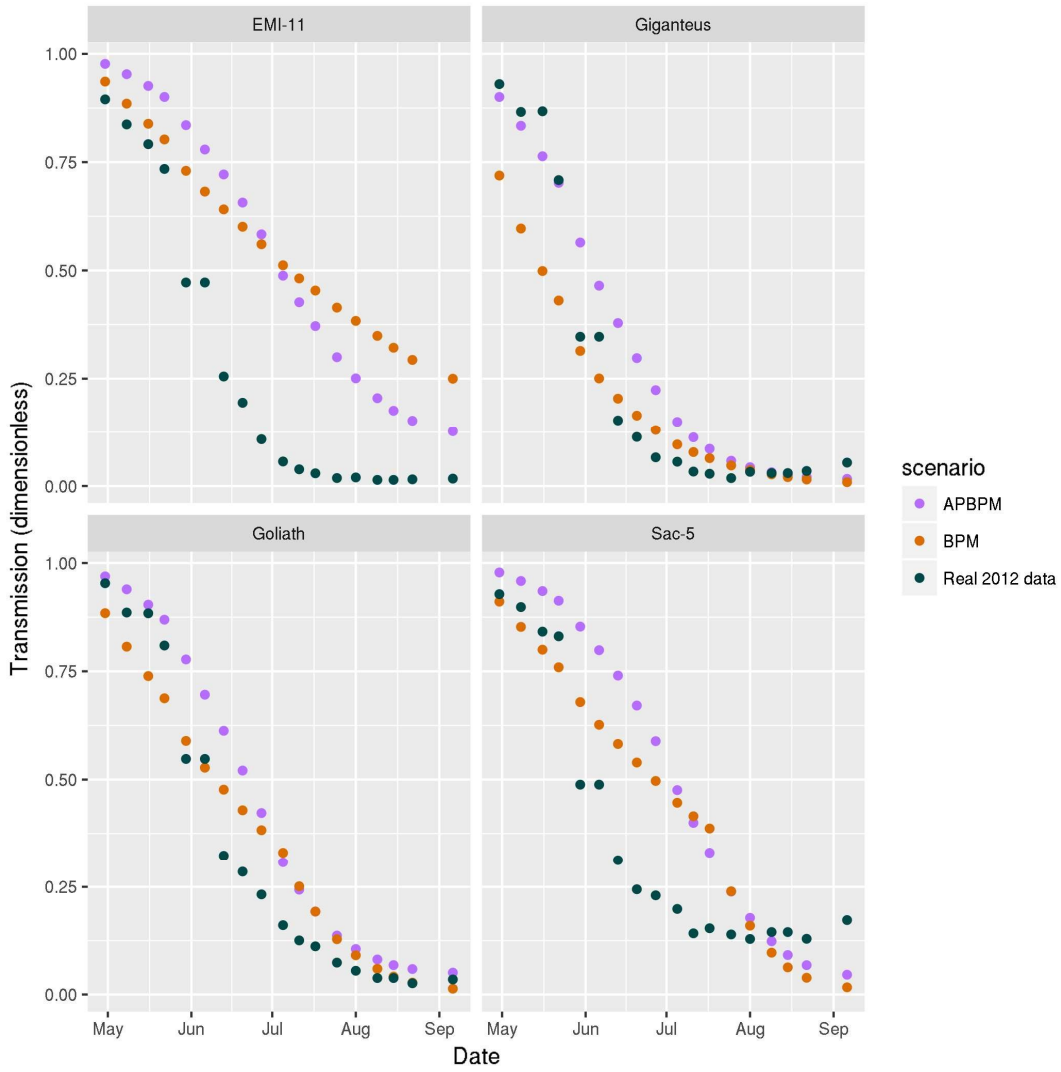

Figure 3.18 Simulated and actual mean transmission values per genotype for the year 2012

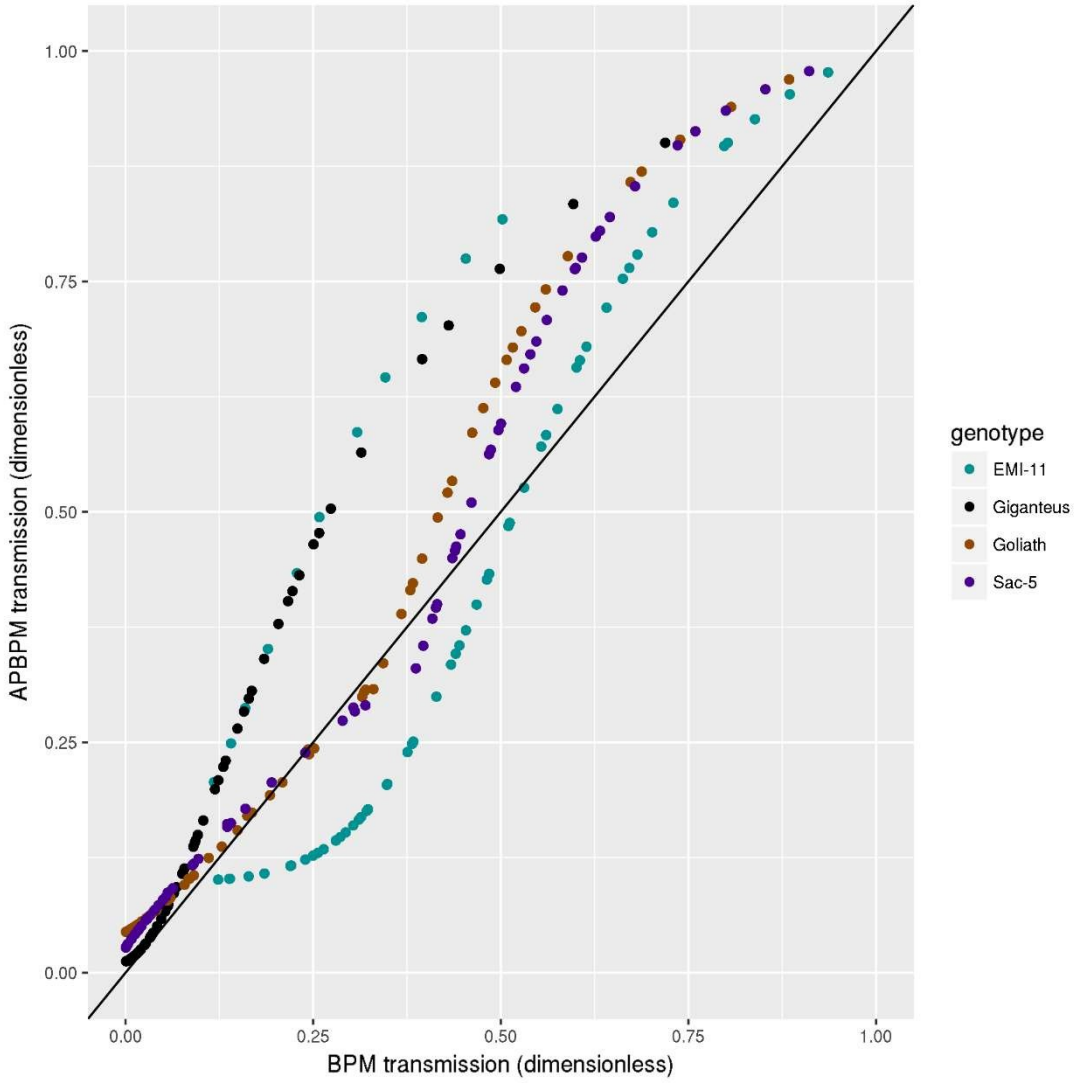

Figure 3.19 Simulated values for transmission generated by APBPM vs those generated by BPM for all years. The main causes of the discrepancy between the two models are the different approach in estimating LER and the use of a single k value per genotype in APBPM, as opposed to two different ones for EMI-11 by BPM. The overall coefficient of determination R^2 between the predictions was 0.899 and the RMSE was 0.109. Table 3.9 shows the R^2 and RMSE for each individual genotype. As transmission was calculated using the simulated value for LAI, the effect of modelling LAI differently (segmented and linear regression in BPM and a sigmoid curve in APBPM) influenced the simulated transmission values.

	Giganteus	Goliath	Sac-5	EMI-11
R^2	0.966	0.976	0.975	0.774
RMSE	0.110	0.080	0.084	0.134

Table 3.6 Coefficient of determination and RMSE of APBPM simulated transmission vs BPM simulated transmission

Genotype	Mean	Standard deviation	Minimum value	25%	50%	75%	Maximum value
EMI-11	0.35	0.33	0	0.04	0.26	0.59	1
Giganteus	0.27	0.34	0.01	0.04	0.08	0.37	1
Goliath	0.37	0.32	0.01	0.1	0.26	0.57	1
Sac-5	0.44	0.29	0.06	0.19	0.34	0.65	1

Table 3.7 Descriptive statistics for the transmission values in the training dataset (years 2011 and 2012).

Genotype	Mean	Standard deviation	Minimum value	25%	50%	75%	Maximum value
EMI-11	0.15	0.29	0	0	0.02	0.1	1
Giganteus	0.25	0.32	0.01	0.04	0.08	0.33	1
Goliath	0.21	0.31	0	0.01	0.04	0.31	1
Sac-5	0.24	0.28	0.02	0.07	0.12	0.32	1

Table 3.8 Descriptive statistics for the transmission values in the test dataset (years 2014, 2015 and 2016).

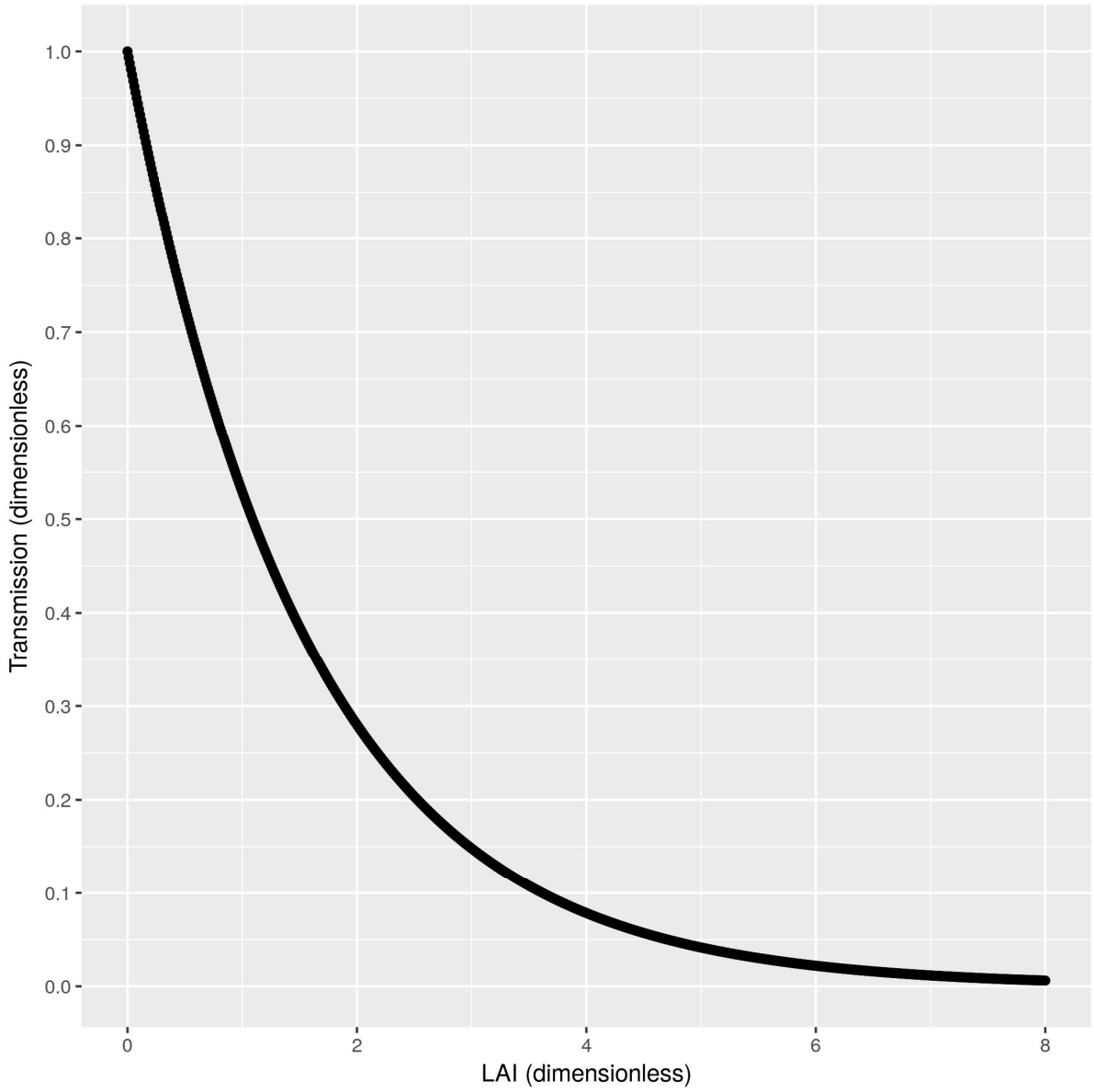

Figure 3.20 An illustration of the relationship between LAI and transmission, as modelled by BPM and APBPM. The transmission values were generated using the transmission part ($Transmission = e^{-k*LAI}$) of the proportion of light equation listed under section 3.2.1. The k value found by APBPM for M. x. Giganteus (0.6365273) was used for the equation k parameter. The graph demonstrates that plants with simulated LAI of above 4 would be considered to be letting through less than 10% of the PAR by the two models, i.e. at that point they will be intercepting more than 90% of the PAR. After that point, the transmission values become so small, that the precise simulated LAI value is irrelevant.

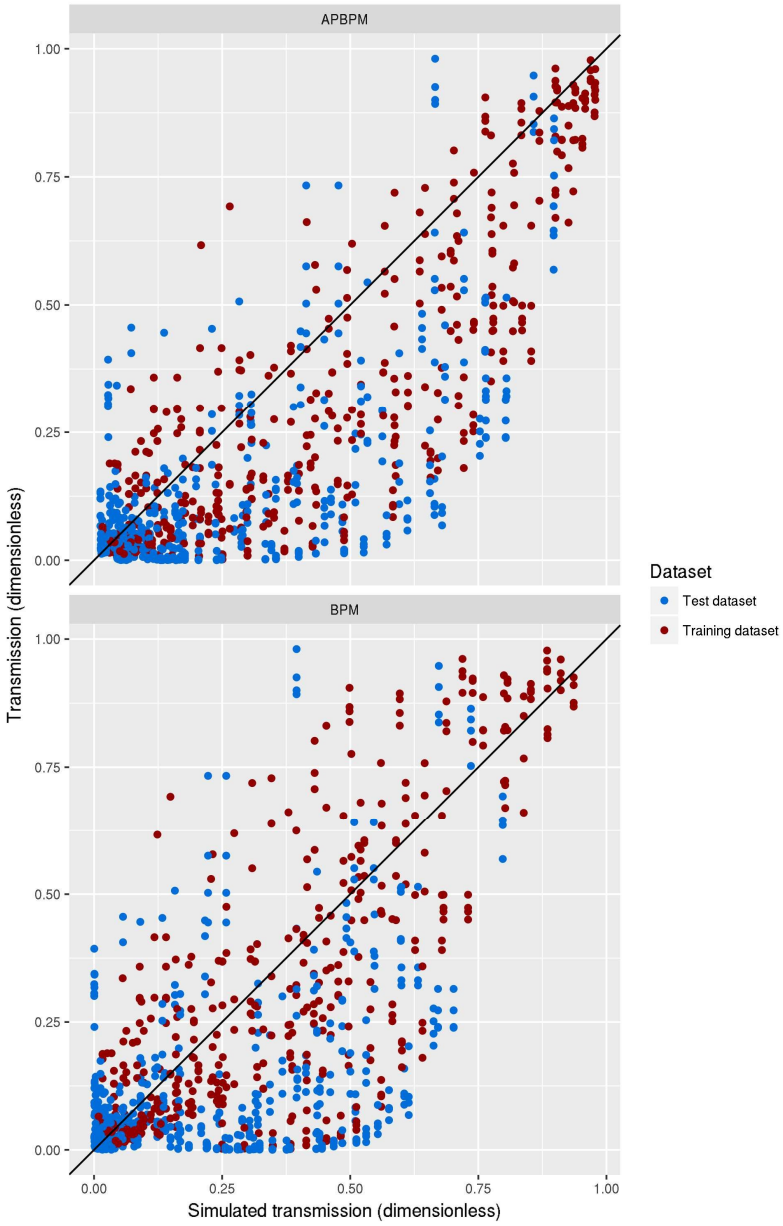

Figure 3.21 Simulated vs actual transmission values for all years. Simulating transmission was a difficult task for the BPM and APBPM. Because of the way they were structured, any errors in the simulation of LAI and the calculation of k, increased the error of transmission simulation. R^2 and RMSE for the two models and datasets are listed in Table 3.9.

	Training dataset		Test dataset	
	R^2	RMSE	R^2	RMSE
BPM	0.612	0.184	0.231	0.232
APBPM	0.721	0.199	0.444	0.223

Table 3.9 Transmission simulation accuracy results for the training dataset and the test dataset.

3.3.3 Cumulative intercepted PAR

The discrepancies between predictions of transmission and Pl had minimal effect on simulation of cumulative intercepted PAR since the start of the growing season. The two models seemed to agree with each other, evidenced by the coefficient of determination R^2 being 0.979 and the RMSE 72.66. Comparing the correlations for each genotype (Table 3.13) showed lower correlation for EMI-11 due to the differences in the models listed so far.

A comparison between the predictions across the models is shown in Figure 3.22. No direct measurements existed of PAR intercepted by the plant, making it impossible to compare simulated values with actual data. This meant that the models could be equally accurate or inaccurate in their simulation of the cumulative intercepted PAR.

3.3.4 Dry matter yield

The RUE values for all genotypes, found by APBPM, were lower than the ones BPM used (Table 3.10). The dissimilarities were caused by the differences in the simulated intercepted PAR values between the two models. Another significant cause was that BPM used data from Goliath, Sac-5 and EMI-11 to estimate a common RUE value for these three genotypes, whereas APBPM calculated the coefficients individually for each genotype.

Despite the differences, the simulated values of dry matter yield generated by the two models followed a similar pattern, as illustrated by the example simulations for year 2011 shown in Figure 3.23, the comparison between the simulated values from the two models in Figure 3.24, and the R^2 and RMSE between the model predictions which were 0.965 and 150.09 respectively. The individual genotype coefficients of determination and RMSE were the same as those for PAR (Table 3.14). This was to be expected as the value of simulated yield depended directly on the simulated intercepted PAR, so any change in the prediction for PAR would be a change in the prediction for yield.

APBPM achieved slight improvements in its prediction accuracy over the training dataset, demonstrated by Table 3.15 and Figure 3.25, which show the prediction performance of the two models against real data. Descriptive values for dry matter yield are shown in Table 3.11 for the training dataset and Table 3.12 for the test dataset.

	Giganteus	Goliath	Sac-5	EMI-11
BPM	2.4	1.66	1.66	1.66
APBPM	1.75	1.32	0.92	0.82

Table 3.10 RUE coefficients for BPM and APBPM

Genotype	Mean	Standard deviation	Minimum value	25%	50%	75%	Maximum value
EMI-11	541	421	17	97	542	906	1300
Giganteus	1168	1005	3	272	832	2126	3254
Goliath	894	835	6	118	936	1442	2633
Sac-5	673	763	1	139	576	954	3565

Table 3.11 Descriptive statistics for the dry weight values in the training dataset (years 2011 and 2012).

Genotype	Mean	Standard deviation	Minimum value	25%	50%	75%	Maximum value
EMI-11	1707	759	551	931	1688	2260	3209
Giganteus	1754	1211	484	794	1462	2179	5759
Goliath	1542	938	258	796	1372	2189	3881
Sac-5	1990	1013	641	1127	1834	2575	4527

Table 3.12 Descriptive statistics for the dry weight values in the test dataset (years 2015 and 2016 only, as no in-season harvests were done in 2014).

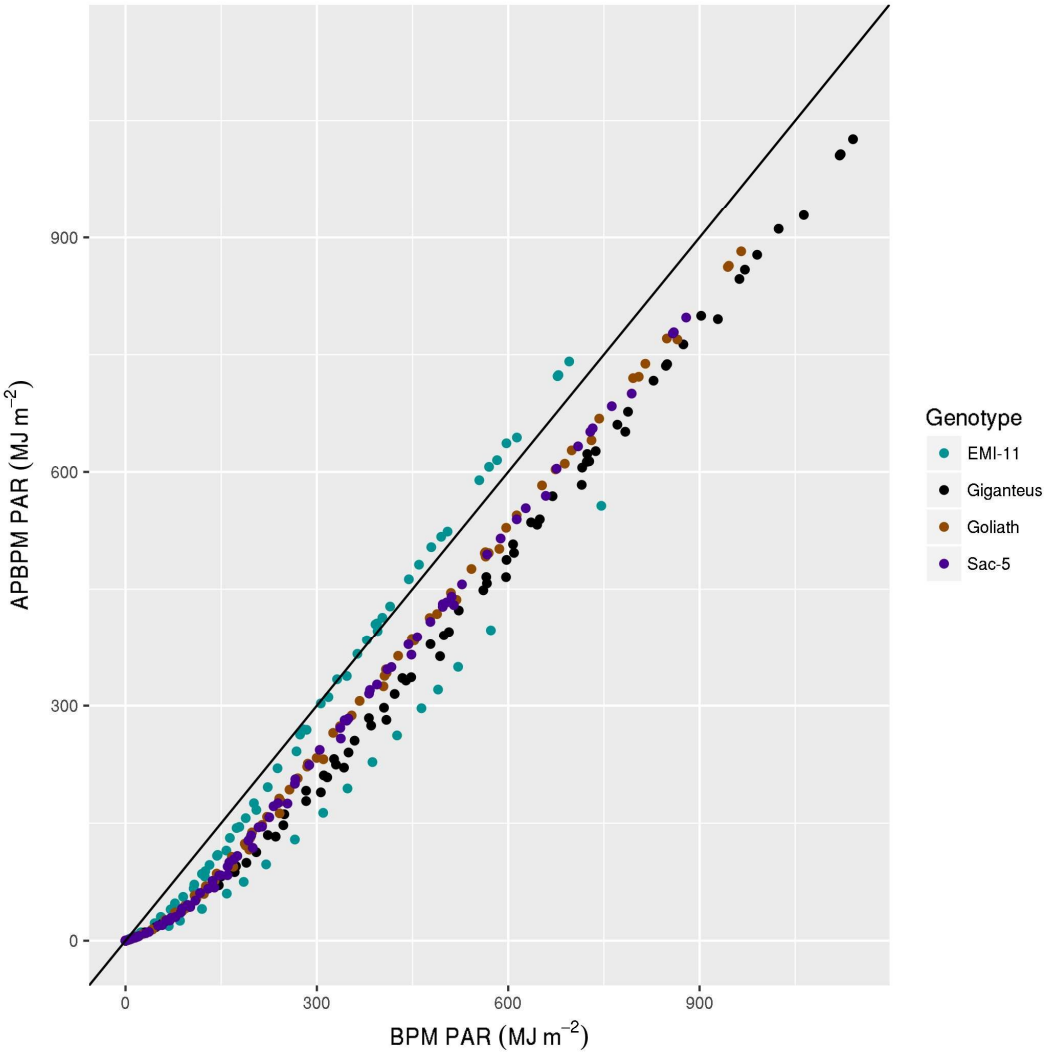

Figure 3.22 Simulated cumulative PAR values generated by APBPM vs those generated by BPM for all years. The overall coefficient of determination R^2 between the model predictions was 0.979 and the RMSE was 72.66. Table 3.13 shows the individual R^2 and RMSE for each genotype. The difference between the two models' simulated values was due to the use of sigmoid by APBPM when simulating LAI. Segmented and linear regression (used by BPM) predicted a faster increase in LAI early in the season, compared to the sigmoid. This lead to BPM simulating higher values earlier in the season for cumulative intercepted PAR than APBPM.

	Giganteus	Goliath	Sac-5	EMI-11
R^2	0.995	0.997	0.996	0.945
RMSE	103	64	64	56

Table 3.13 Coefficient of determination for simulated intercepted PAR values from APBPM vs BPM.

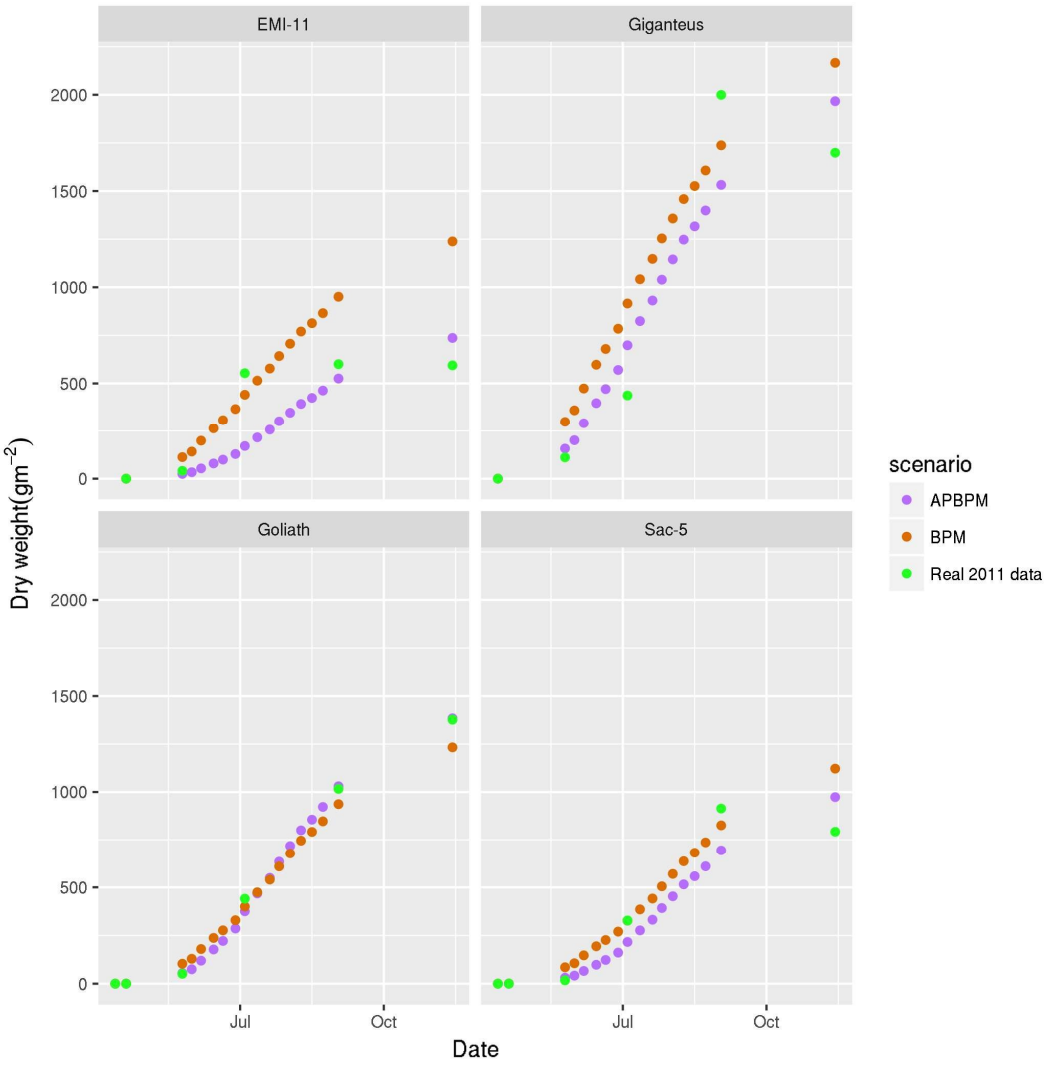

Figure 3.23 Simulated and actual mean yield values per genotype for the year 2011. The data points furthest to the right denote the last harvest done in 2011, which was in December.

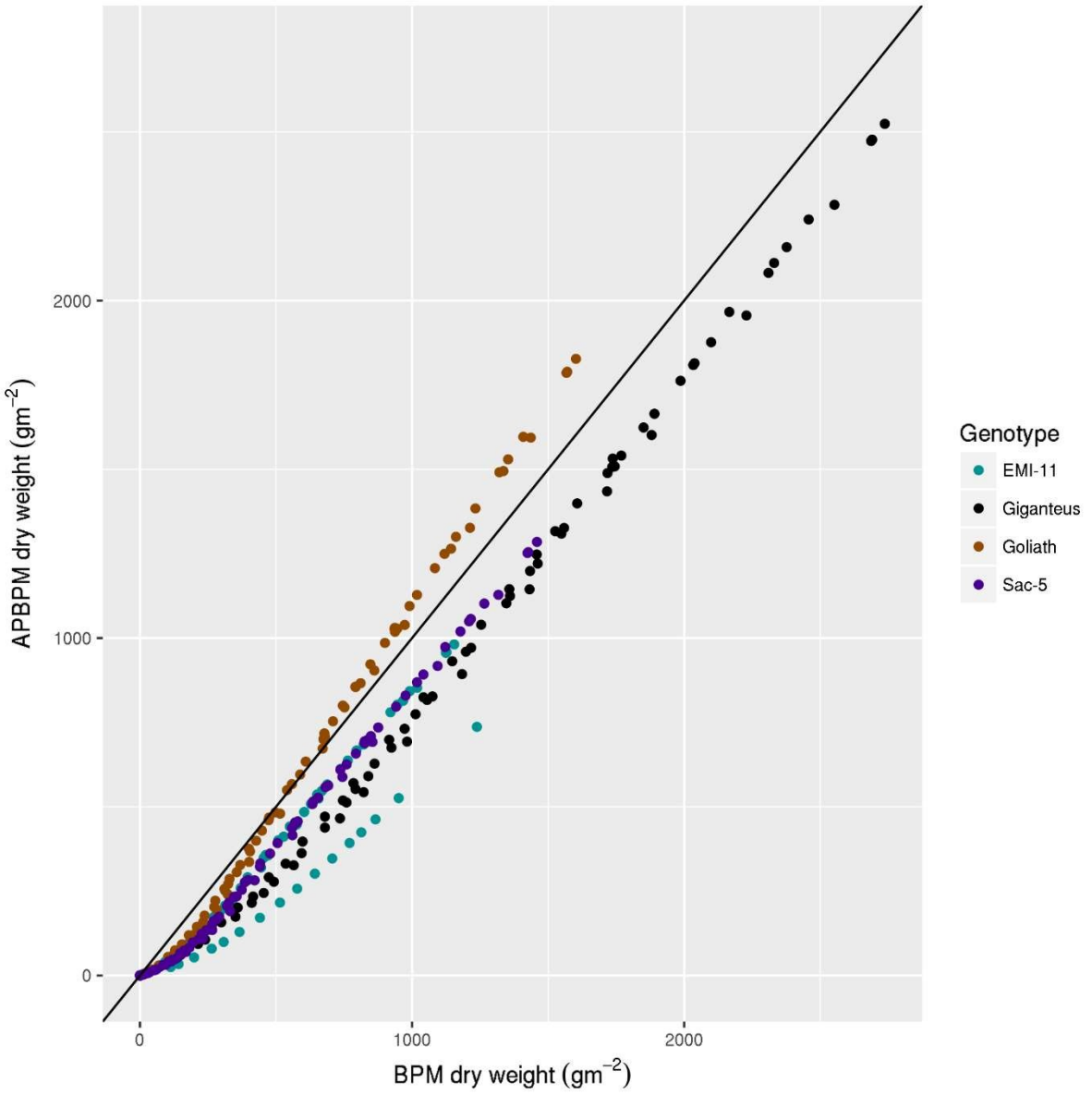

Figure 3.24 Simulated yield values generated by APBPM vs those generated by BPM for all years. The overall coefficient of determination R^2 between the model predictions was 0.965 and the RMSE was 150.09. Table 3.12 shows the R^2 and RMSE for each individual genotype.

	Giganteus	Goliath	Sac-5	EMI-11
R^2	0.995	0.997	0.996	0.945
RMSE	223	81	119	140

Table 3.14 Coefficient of determination and RMSE for simulated dry matter yield values from APBPM vs BPM

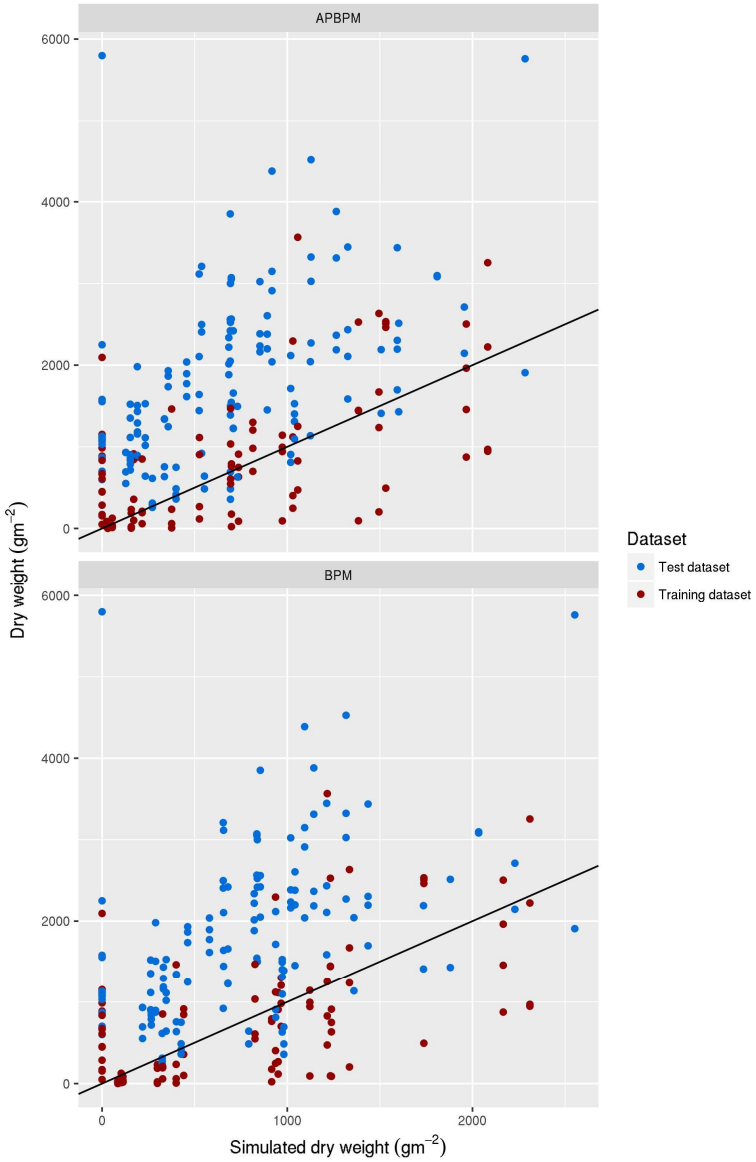

Figure 3.25 Simulated vs actual dry matter yield values for all years. The training data contained harvest data from years 2011 and 2012 – plants were not mature at the time as they were only 2 and 3 years old. In the harvests captured by the test dataset (during years 2015 and 2016), the plants were mature. R^2 and RMSE statistics for these plots are shown in Table 3.15.

	Training dataset		Test dataset	
	R^2	RMSE	R^2	RMSE
BPM	0.329	692	0.256	1340
APBPM	0.369	673	0.260	1408

Table 3.15 Dry matter yield simulation accuracy results for the training dataset and the test dataset. Figure 3.25 shows prediction and actual values for the training and test datasets.

3.4 Discussion

The results from the APBPM simulations compared well with the BPM simulations in terms of predicted yield and PAR. The different RUE parameter values compensated for any discrepancies in the simulated PAR data, ensuring that the two models produced similar yield predictions. The sigmoid approach to modelling LAI improved APBPM's fit to LAI and transmission in both the training and the test dataset. It was also a more objective approach to modelling LAI, that eliminated the need to manually remove plateau data points, as was done in the BPM publication. This method was not perfect as it failed to predict late season decline of LAI due to senescence (example from 2014 shown in Figure 3.15). If APBPM was trained on LAI data with the decline phase, the sigmoid would have likely underestimated the top (plateau) yield, while trying to fit to the late season data. However, due to the relationship between simulated LAI and transmission (Figure 3.20), if the plateau LAI value is above 4, the plant would be modelled to be intercepting most of the light. Thus, the impact on accuracy for cumulative intercepted PAR and yield would have been insignificant.

A better approach for modelling LAI used in the MISCANFOR model described by (Hastings *et al.*, 2009). There the start of leaf senescence was modelled against cumulative degree days, and LAI was allowed to decay at a constant rate from that point onward. As the aim of the automation procedure was to reproduce BPM as closely as possible, and the MISCANFOR approach would have required more data and increased the model complexity, it was decided to use the sigmoid approach instead.

The simulations of EMI-11 genotype were the point where the two models disagreed the most, particularly for light interception (Figure 3.19), PAR (Figure 3.22) and dry matter yield (Figure 3.24). The different growth that the genotype exhibited in the two-year training dataset made it difficult to reproduce BPM predictions. While a different k parameter could have been fitted to each year, the subjective decision to treat the two years separately would have been impossible to reproduce in the automated procedure. If each year was treated separately by default, a decision had to be made which parameter to use when simulating an unseen year. This was particularly true in ambivalent cases, for example, the model was trained on years 2014 and 2016 and had to simulate 2015. To avoid similar conundrums and simplify the automation of the parameterisation, k parameters were calculated using a more objective approach, combining all data for the genotype from the training dataset together.

The work in this chapter uncovered an interesting question, on whether the value of k does change between years. Some difference in the relationship between LAI and transmission in 2011 and 2012 could be seen on Figure 3.5 and Figure 3.16. While this was not investigated further, if a future study confirms that this is the case, that could influence greatly how mechanistic models are parameterised and how they could be designed to generalise for future growing seasons.

Simulations over the test dataset were significantly less accurate than those over the training dataset. While there is always an increase in error over unseen data, an additional factor may have been plant maturity. During 2011 and 2012 plants were most probably still immature, compared to later years. The overall increase of LAI (Figure 3.14) and yield (Figure 3.25) in the test dataset, compared to the training data suggested that the plants had not yet reached their peak yield in the earlier years. Also, possible drought effects may have also influenced the growth of the plants in the growing seasons covered by the test dataset. The lack of drought modelling in BPM and APBPM could be an additional cause of loss of accuracy. Including some data from later years into the training dataset for APBPM could have improved its performance over the test dataset. This was done in work presented in later chapters, where its prediction performance was compared to that of other models. However, it was not done here, as it was outside of the scope of this experiment, which was to demonstrate similarity between APBPM and BPM. Training on different data would have resulted in bigger differences between the simulations from the two models.

Based upon the demonstrated similarity between the two models, it can be concluded that the automated parameterisation procedure produced a close, though not exact, reproduction of BPM. Statistical comparison between coefficients was not needed, which meant that APBPM could work on genotype data from all plots, instead of fitting the parameter values to the noisier data from each individual plot. Subjective decisions (e.g. removing plateau LAI measurements) could not be automated, which required changing the modelling approach. Improvements to the model were made, to allow it to be parameterised using reduced datasets, which was done later in the project. The impossibility of automating the parameterisation as described in the BPM publication, as well as the different requirements for APBPM resulted in a procedure that allowed quick parameterisation of the model using an arbitrary datasets compatible with the model. Thus, data from later years could be added

to the training dataset or the model could be trained to simulate new genotypes. The procedure reduced the time it would take to parameterise the model, from at least several days, when following the procedure for parameterising BPM to 10 seconds, when done automatically.

Chapter 4: Screening of machine learning methods

This chapter describes the development of machine learning models using supervised learning algorithms, for predicting *Miscanthus* yield. Two types of models were examined: simple models, which predicted yield directly from all phenotypic and meteorological variables, and complex models, which were comprised of several machine learning submodels to predict phenotypic variables from meteorological data, and a final submodel to predict yield. The structure of a complex model, which meant the arrangements of submodels and data flow from one to the other was another problem that was examined. Two solutions were presented – a naïve structure, and a structure optimised with genetic algorithms (GA). All models were cross-validated, and their accuracies were compared with APBPM's accuracy. The most accurate models were used in the experiment described in chapter 5. From the simple models these were random forests, GBM and k-NN, and they outperformed APBPM. The naïvely-structured complex model performed better than the one optimised with GA, but both were better than APBPM.

4.1 Introduction

Machine learning approaches have a wide application in crop modelling. There have been numerous studies on predicting crop yield from various input variables. Artificial neural networks (ANN) are a popular model choice for this task. One study looked into predicting soybean yields from soil parameters (pH, organic matter, phosphorus, potassium, elevation, etc.) using feedforward neural networks (Drummond, Joshi and Sudduth, 1998). Other studies using a similar method expanded the number of variables by adding meteorological data, crop management information (nitrogen application) (Liu, Goering and Tian, 2001), as well as more landscape and soil chemical properties (Miao, Mulla and Robert, 2006), for predicting corn yield and grain quality. The ability of ANN to predict yield for various crops including cotton, wheat, sugarcane, rice and others using meteorological and soil variables has been demonstrated in a later study (Dahikar and Rode, 2014).

Other models have also been applied to the problem, including the C4.5 decision tree algorithm (Quinlan, 1993), for soybean yield prediction using soil physical and

chemical properties (Canteri *et al.*, 2002). Another study applied linear regression, M5-Prime (Quinlan, 1992) (algorithm for building regression trees), support vector regression (SVR), k-nearest neighbour (k-NN), and ANN for predicting yields from multiple crops using meteorological data, planting area, and amount of applied water for irrigation (Gonzalez-Sanchez, Frausto-Solis and Ojeda-Bustamante, 2014). It found the M5-Prime to produce the best predictor among the tested models. A review of papers focusing on the usage of machine learning for predicting biomass from remote sensing data (Ali *et al.*, 2015) lists several studies that used ANN, support vector machines (SVM), linear regression, and k-NN for predicting biomass. A more recent study into modelling drought from remote sensing data (Park *et al.*, 2016) used cubist, a proprietary rule-based model (Quinlan, 1992) based on C4.5 and M5-Prime (Minasny and McBratney, 2008), random forests (RF) and boosted regression trees. It modelled standardized precipitation index (SPI (McKee, Doesken and Kleist, 1995), an indicator for meteorological drought, and yield in corn and soybean and found that random forests produced the best prediction performance.

Machine learning models have been used for modelling other crop related variables as well. SVM were shown to outperform ANN-based models in predicting rice blast disease in rice from environmental data (Kaundal, Kapoor and Raghava, 2006), and performed well in classifying diseases in sugar beet from remote sensing data (Rumpf *et al.*, 2010). They have been used for crop classification from remote sensing data (Mountrakis, Im and Ogole, 2011; Zheng *et al.*, 2015), alongside ANN, random forests, decision trees, etc. (Ali *et al.*, 2015). Leaf area index (LAI) has been modelled from cumulative photosynthetically active radiation (PAR) absorbed by the plant using ANN (Dunea and Moise, 2008), and from remote sensing data in numerous studies (Ali *et al.*, 2015), using ANN and SVM. Another study used an ANN, a decision tree and a naïve Bayes classifier to model the seed quality in cotton (Jamuna *et al.*, 2010).

Genetic algorithms (GAs) are another tool that is finding more and more applications in crop modelling. GAs are a heuristic technique for optimisation (Oh *et al.*, 1999) inspired by evolutionary processes in nature. This method works by “evolving” a population of candidate solutions to a problem through a process that mimics natural selection (Mitchell, 1996). Each solution is represented by a chromosome, which

could be a string of ones and zeros ("bits") and describes the solution. The exact representation of the solution into a chromosome form is problem dependent. Each solution has an associated fitness value to it, which describes how favourable it is. Several evolution-inspired operators are applied to the population of solutions: selection, crossover, and mutation. Selection chooses the best solutions in the population based on their fitness values. Crossover combines two solutions using a process that emulates chromosomal crossover in nature. Mutation changes the value of a random gene on the chromosome. In the example where chromosomes are strings of bits, mutation would change the value a random bit (gene) from 1 to 0 or vice versa. By repeatedly applying these operators, GAs generate new generations of solutions with improved fitness, which continues until the point where the algorithm cannot generate any better solutions. There have been several studies where GAs have been used for optimising parameter values for crop models (Fang, 2003; Dai *et al.*, 2009; Klein *et al.*, 2012; Waongo *et al.*, 2013). The general approach they follow for applying GAs to the problem is to generate populations of solutions (sets of parameter values) and improve these solutions through the application of the genetic algorithm operators, with respect to the crop model's performance in the required task.

This chapter describes the training and cross validation of a wide variety of supervised machine learning models for the regression task of biomass yield prediction. Machine learning methods used in this chapter are described in section 2.2. First, dry matter yield from the ABR61 trial dataset was modelled through standard machine learning models using all available variables in the data. Then, more complex models, called compound models, were developed. They were formed of several machine learning submodels, and were more similar in functionality to the APBPM model, described in chapter 3, because they were driven by meteorological data and predicted phenotypic data. The structure of one of the compound models was optimised using GAs. Finally, model accuracies were compared to establish each model's prediction performance.

A selection of the best models established here were used in the following chapter, for determining the importance of variables and time of measurements in model

training data. Importance was determined by removing variables and data points from the training dataset and testing the influence on prediction error. For this reason, the most accurate models discussed in this chapter were chosen, as they made the best use of the training data and therefore their modelling error reflected the amount of useful information within the training data.

4.2 Materials and methods

4.2.1 Data overview

The dataset used for training and testing models within this chapter came from the ABR61 trial. This was the same dataset that was used in the previous chapter for parameterising APBPM and validating its predictions. The models described here accepted a wider range of variables as input compared to BPM and APBPM, both for training and prediction. Table 4.1 presents a comparison between the input data for BPM, APBPM, and the models developed here. Meteorological measurements (total PAR on the day (TPAR), rainfall, degree days) were provided to the models as cumulative values since plant emergence. Phenotypic and meteorological variables are explained in detail in sections 2.1.2 and 2.1.3. Metadata variables (genotype, the

Model	Genotype	Plot row	Plot column	Day of year	Year	TPAR	Rainfall	Degree days	Stem count	Canopy height	LAI	Flowering score	Transmission	Dry weight	Plant emergence
BPM	✓*	×	×	×	✓*	✓	×	✓	✓*	×	✓	×	✓	✓	✓*
APBPM	✓*	×	×	×	×	✓	×	✓	✓*	×	✓	×	✓	✓	✓*
Machine learning models	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓*

Table 4.1 Comparison between the variables available in the ABR61 dataset used in BPM, APBPM and the machine learning models described in this chapter.

✓ - the model used the variable

✓ - the model used variable indirectly (genotype was used by BPM/APBPM to assign parameter values to each genotype, BPM used year to assign different parameters each year where needed (k parameter for EMI-11), stem count was used to calculate LAI). For machine learning models, plant emergence was used to calculate cumulative rainfall, degree days and TPAR since emergence. BPM and APBPM used plant emergence as a starting point of their simulation.*

× - the model did not use the variable

row and column number of the plot in the field, day of year and year) contained data related to each data point. They defined the genotype of the plant being measured (genotype), the location in the field of the plot that contained the plant (plot row and column), and date of the measurement (day of year and year). Each data point was a single date where measurements when measurements were taken. The value for each variable described the current measurement at the time – Genotype, row, and column never changed; day of year and year reflected the current date; TPAR, rainfall and degree days were the cumulative value since emergence on the day of measurement; the values for the phenotypic variables were what was measured on that date.

4.2.2 Model categories, training and testing

The models described in this chapter fell within two categories depending on their approach to modelling and the training data they accepted: “simple models” and “compound models”. The approach in simple models was to use a single machine learning algorithm to train a model that predicted yield directly from all available variables. Compound models incorporated several machine learning submodels that predicted various phenotypic variables as intermediate steps before predicting yield. The different modelling approach between the two categories dictate different model structures which are discussed in sections 4.2.3 and 4.2.4. Despite the differences in approach and input data, models from both categories operated in the same way, they accepted input data which was used to produce predictions of dry weight, as illustrated in Figure 4.1.

The models were the result of the application of a common training procedure on the training dataset. An overview of the procedure is shown in Figure 4.2. The approach involved passing all available training data to an algorithm tasked with developing a model that produced the most accurate yield predictions over that data. The actual algorithm depended on the model category, with off-the-shelf machine learning algorithms used directly in simple models’ training, and more complex custom algorithms used for compound models.

Models were then compared based on their ability to predict biomass yield. The whole training and test procedure is illustrated in Figure 4.3. To avoid any overfitting effects, the prediction accuracy was measured using a test dataset, which was fed through each model, allowing it to generate yield predictions. The models' accuracies were calculated using root mean square error (RMSE) and R^2 (the square of Pearson's correlation coefficient) over the predictions generated by the models and the real dry weight values. Section 2.2.7 describes the two statistics in detail.

The data in the test dataset was not present in the training data and was not made available to the model during training. The model accessed the test data only when it generated predictions over it. Crop models are usually trained on historic data and used to predict biomass yield in future years. To reflect this expectation in the experimental results, the training and test datasets contained data from different years taken from the ABR61 data. Thus, if the training dataset contained data from the years 2011 and 2015, the test dataset would have contained data from 2016. The exact testing procedure differed between the simple and the compound models and is covered in more detail in the following subsections.

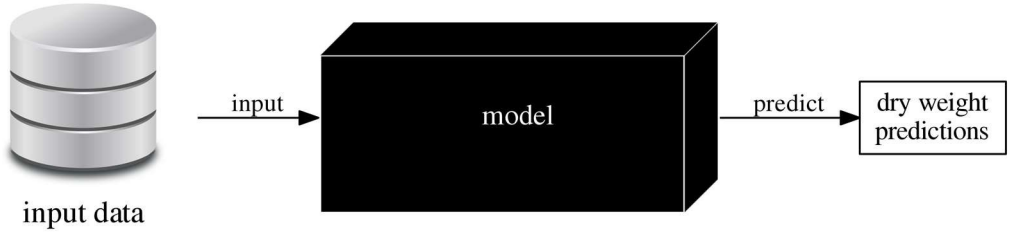

Figure 4.1 Basic machine learning model operation - the model used the provided input data to produce dry weight predictions.

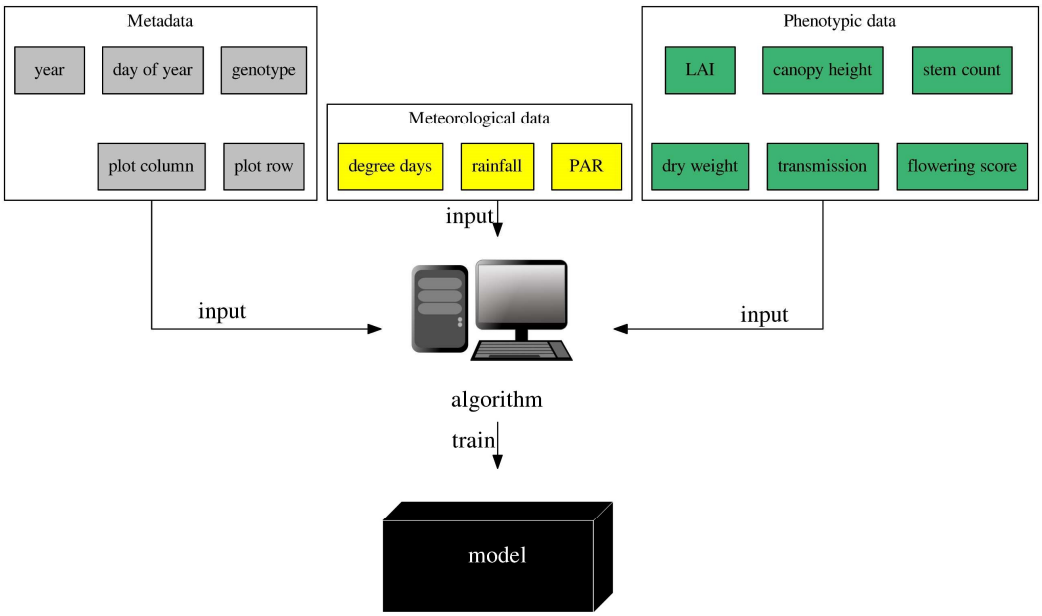

Figure 4.2 Overall model training procedure. All available training data (metadata, meteorological and phenotypic data, including dry weight measurements) was provided as input to an algorithm which produced a model. The input data had the same format as illustrated in Table 2.2 – phenotypic data contained all weekly or fortnightly measurements (depending on the season), and the meteorological data were cumulative from plant emergence to the day of measurement.

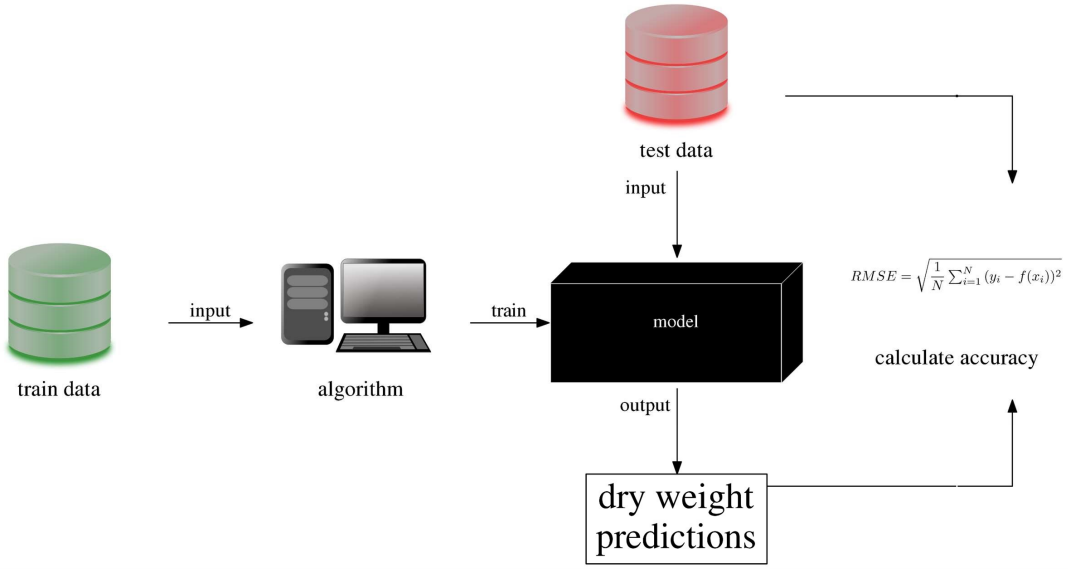

Figure 4.3 Training and test procedure for machine learning models. Training data was used by the training algorithm to train the model. The model was then applied to the test dataset and produced dry weight predictions for it. Finally, the predicted values and the real measurements from the test data were passed through the RMSE equation to calculate the prediction accuracy of the model.

4.2.3 Simple machine learning models

As mentioned in the previous section, simple models were single machine learning models that were trained to predict yield using metadata, meteorological and phenotypic variables. Once trained, they followed a common procedure for generating dry weight predictions, illustrated on Figure 4.4. For each data record, they used all available measurements to produce a dry weight prediction.

A wide range of machine learning algorithms were used to train the models, to identify the best approach to modelling *Miscanthus* data among them. The algorithms used in this category were: linear regression (or linear model (LM)), ANN, SVM, gradient boosting models (GBM), kNN and random forests (RF). Each model's accuracy was established using cross-validation and calculating RMSE and R^2. Then the accuracy of APBPM, described in the previous chapter, was calculated using the same method. Finally, the machine learning models and APBPM were compared using their prediction accuracies. The R script used to train the model can be found in section 8.1 in the appendix.

Model training

All models were trained with the "train" function from the "caret" R-package (Kuhn, 2016), which was applied to functions from a variety of R libraries (R Core Team, 2017), that provided procedures for training different machine learning methods. Table 4.2 lists all machine learning algorithms, their corresponding method name parameter in "caret" and the underlying R function.

Model name	Method name	Tuning parameters	Underlying function	Reference
Linear regression (LM)	lm	intercept	lm	(R Core Team, 2017)
ANN	nnet	size, decay	nnet	(Venables and Ripley, 2002)
SVM	svmLinear3	cost, loss	LiblineaR	(Helleputte, 2017)
GBM	gbm	n.trees, interaction.depth, shrinkage, n.minobsinnode	gbm	(Ridgeway, 2015)
k-NN	kknn	kmax, distance, kernel	kknn	(Schliep and Hechenbichler, 2016)
RF	rf	mtry	randomForest	(Liaw and Wiener, 2002)

Table 4.2 Caret method used for each model. The function found the optimal values for each parameter under the tuning parameters column and trained the model using the underlying function.

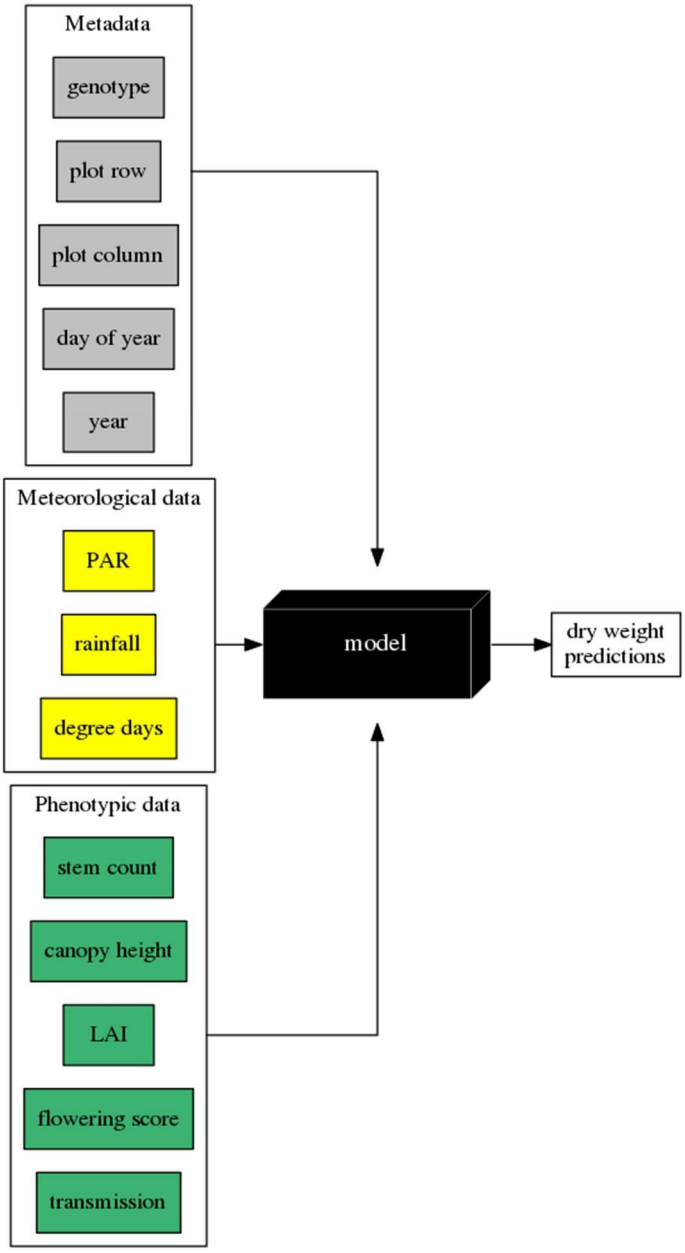

Figure 4.4 Structure of simple machine learning models. A single machine learning model takes metadata, meteorological and phenotypic measurements as inputs and outputs yield predictions. Metadata refers to data related to the individual data point: type of measured plant (genotype), location of the plot in the field (plot row, plot column), and date (day of year and year). The model in the box was the model created by the application of a machine learning algorithm on training data. The machine learning algorithms used in this chapter were: linear regression, k-nearest neighbour, random forests, artificial neural networks, support vector machine and generalised boosted models.

The aim of the train function was to find the optimal values for the model parameters provided at training by the user, e.g. number of trees in random forests or number of neurons in the hidden layer for ANN. These tuning parameters were different for each individual model. The "train" function determined the optimal tuning parameter values for each model, by defining numerous sets of tuning parameter values and evaluating their performance. A random sample of records from the training data was withheld, and the rest was given to the model for training. The model was then used to predict the withheld samples. This process was repeated multiple times for each parameter set, and the average prediction accuracy was calculated across the withheld predictions, which gave the average performance for each set of parameter values. The optimal parameter set was the one that gave the best prediction accuracy. After tuning the parameters, the final model was trained over the whole training dataset.

The "train" function provided multiple versions of some models, depending on which implementation of the parameterisation procedure was applied. The user could choose between versions using the method parameter. The method used for each model is listed in Table 4.2 as well as the tuning parameters. The intercept parameter for linear regression was a logical variable denoting whether the linear regression model would fit an intercept term or not. The "nnet" fitted a single-hidden-layer neural network to the data. The size parameter described the number of neurons in the hidden layer. The decay was the weight decay λ, which controlled the influence of the penalty term of the network weights $\mathrm{J}(\theta)$ over the loss function. The "svmLinear3" method trained SVM using L_2 regularisation, and chose between using L_1 or L_2 loss function (defined by the loss tuning parameter). The cost parameter was the weight of the regularisation term.

The "gbm" method trained a stochastic gradient boosting model, a modification of the original gradient boosting model that uses a random sample without replacement for training at each iteration, instead of the whole dataset (Friedman, 2002). The number of trees (and iterations) was determined by the n.trees tuning parameter, and interaction.depth restricted their size. The size of the trees J limited the interaction level between the variables in the model, to no greater than $\mathrm{J}-1$.

For interaction.depth=1 ($J = 2$) the fitted model was an additive model with no interaction between the variables, and by increasing the depth, more variables were allowed to interact with each other (Hastie, Tibshirani and Friedman, 2009). The shrinkage parameter was a regularisation term that scaled the contribution of each tree to the final prediction by a factor in the range between 0 and 1. This controlled the learning rate of the algorithm. Finally, n.minobsinnode defined the minimum number of observations in each terminal node. This acted as a stopping condition for the growth of a tree, when branching on a node would have resulted in the new terminal nodes containing less than the minimum observations.

The "kknn" method fitted a weighted k-nearest neighbour model, where on prediction each observation was assigned a weight, depending on its distance from the query point, with higher weights assigned to closer observations. The kmax tuning parameter governed the maximum number of neighbours used for making the prediction. For a given query point the model calculated the Minkowski distance between it and each observation. The distance parameter was the Minkowski coefficient (q). It determined the type of distance measure used, with $q = 1$ defining Manhattan distance, $q = 2$ specified Euclidian distance, etc. (Zhang, 2012). A kernel function then calculated the weight of the observation depending on its distance from the query point. The choice of kernel function was determined by the kernel parameter.

The "rf" method trained a random forests model using the underlying "randomForest" function. The only tuning parameter for it was mtry. When growing each random-forest tree, the random forests algorithm randomly choose a subset m of all variables on each split, and picked the variable to split on from that subset (Hastie, Tibshirani and Friedman, 2009). The parameter mtry was the size of the subset.

Bagged models

A bagged version of each simple machine learning model listed so far was developed, to test whether bagging improved its prediction capability. The only exception was random forests as it already was a bagged model. The training procedure shown on Figure 4.3 was modified to enable training multiple instances of the same model, on

different versions of the training data. Figure 4.5 shows the training procedure for the bagged models. The first step was to draw random samples with replacement from the training dataset of size equal to the number of instances in the training data. This was repeated n number of times, giving n random samples, where n was the number of models in the bag of models. This number was different for each machine learning method and depended on the time it took to train all n models. The models LM, KNN, and GBM had their n set to 100. For ANN that number was 10 and for SVM 5, as these two algorithms were slow to train, and training 100 of these models was not feasible. Each random sample was provided for training a single model. The output of the training procedure was a bag of models of size n.

The prediction procedure of the bagged versions of the models also differed from the standard approach shown in Figure 4.4. As with the standard machine learning models, predictions were always made on test data which was kept separate from the training data. For a given test dataset, the data was passed to each model to generate predictions. A mean prediction was calculated with the prediction values from each model. This mean prediction was then taken as the final prediction from the bagged model. The approach is illustrated in Figure 4.6. The mean was used as this was what Breiman used for random forests regression (Breiman, 2001a) and this approach sought to replicate his methodology, with a different base learner. Median could also be used, which could lower the influence of models with outlier predictions.

Cross-validation and model accuracy

The models described so far were compared on their prediction accuracy. The models were trained and their accuracies were calculated using a cross-validation approach. The data from ABR61 was divided into groups according to the year the measurements were made. When training, the models could only accept instances where all measurements were available, so all records that did not satisfy that requirement were removed. This meant that years 2012 and 2014 were not used with the simple machine learning models as they did not contain any dry weight measurements made during the growing season.

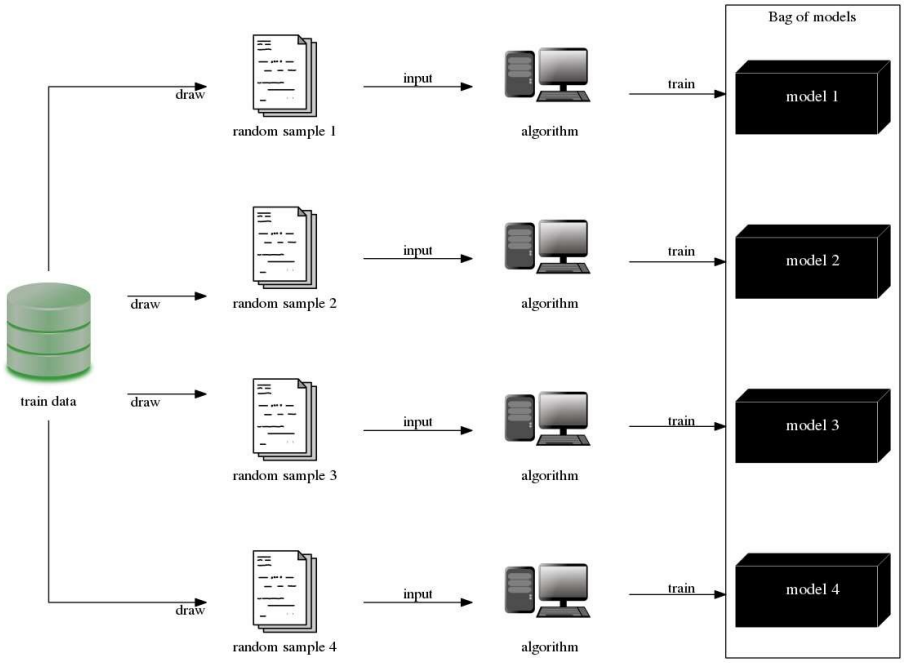

Figure 4.5 Training procedure of a bagged model. Random samples were drawn with replacement from the training dataset resulting in n number of random samples. Each random sample served as training data for a single model. The training procedure produced a bag of models of size n which formed the final bagged model.

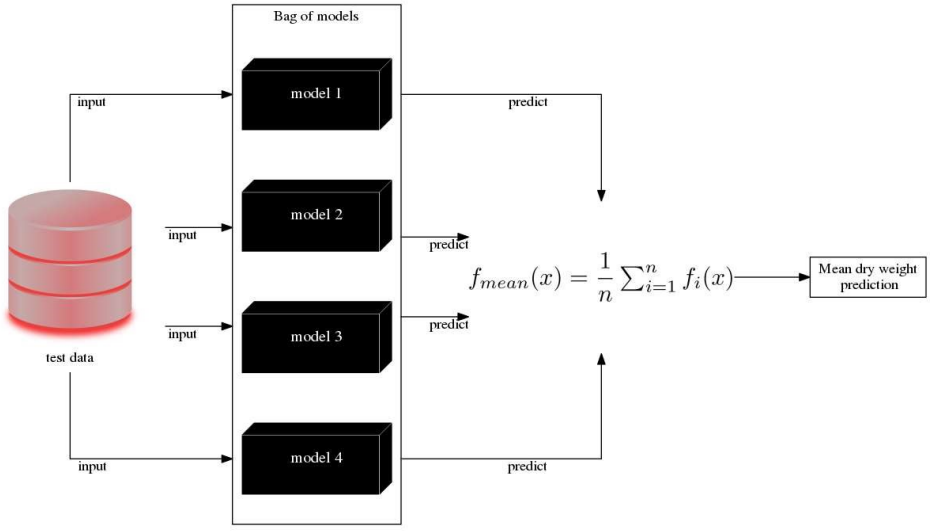

Figure 4.6 Predicting using a bagged model version. For a given test dataset, the data was provided to each model from the bag, which in turn generated its own prediction. The predictions from all models were used to find the mean prediction, which was then output as the final prediction from the bag of models.

A K-fold cross-validation was used to train and test the models, where the dataset was divided by the years that still remained (2011, 2015 and 2016). The content of each cross-validation part and its index number k are summarised in Table 4.3. This division of data by years strategy was employed to simulate a realistic crop modelling scenario, where models are asked to predict yield for the current year but are trained on data from previous years.

CV part	2011	2015	2016
k	1	2	3

Table 4.3 Years from the dataset in each cross-validation fold and the corresponding fold index k.

Models were tested against each cross-validation part after training them using the rest of the data. For example, for $\mathrm{k} = 1$, the training data consisted of parts 2 and 3, and the model predicted yield for part 1 (the test dataset). This is illustrated in Figure 4.7. The same procedure was repeated for $\mathrm{k} = \{2, 3\}$. Finally, the predictions from all folds were pooled together and RMSE and R^2 were calculated.

For APBPM, the cross-validation procedure was the same, except that no records were removed from the ABR61 dataset. This was because it could use instances with missing measurements for training its various modules. Testing (yield prediction) was still done on the years 2011, 2015, and 2016, as they were the ones with yield measurements during the growing season, allowing the calculation of RMSE and R^2. For each fold, the training data was simply the rest of the ABR61 data. For example, for fold 2, where the test data consisted of measurements from the year 2015, years 2011, 2012, 2014, and 2016 were used for training.

Comparison of standard and bagged model versions

The standard models (KNN, ANN, GBM and LM) were compared to their bagged versions by testing for differences between the errors of the standard and the bagged model. Root square errors (RSE) for each individual prediction were first calculated using the formula below:

$$\mathrm{RSE_i} = \sqrt{\left(\mathrm{y_i} - \mathrm{f(x_i)}\right)^2}$$

Where:

- y_i was an individual yield measurement
- $f(x_i)$ was an individual prediction

The distribution of RSE for each model was then tested for normality using the Shapiro-Wilk test (Shapiro and Wilk, 1965) using the "shapiro.test" function in R (R Core Team, 2017). As they were found to be non-normal, it was impossible to use a t-test because of its assumption of normality. A Wilcoxon signed-rank test (Woolson, 2007) was used instead, for testing for statistically significant differences between the distributions from which the error values of the standard and bagged model versions were drawn. This was done using the "wilcox.test" function in R. Unlike the t-test, the Wilcoxon signed-rank test does not assume normality in the two variables. It also assumes dependency of samples, which is appropriate as pairs of predictions of the two models on the same data point were compared. Two one-tailed Wilcoxon signed-rank tests were performed for every method to determine whether the mean of the error distribution of the standard model was larger or smaller, compared to the mean of the error distribution of the bagged model and how statistically significant was that difference.

4.2.4 Compound models

The drawback of the simple machine learning models when compared to mechanistic models like BPM, MISCANMOD and MISCANFOR was that while the latter were driven purely by meteorological data, the machine learning models required the phenotypic data as well to produce a prediction. This made them less applicable to scenarios where phenotypic data is not available, like predicting production in future seasons, or trying to predict potential production in another climate. To correct for this, another type of models called compound models were developed.

Compound models were models made up of several machine learning submodels, each of which predicted a single phenotypic variable (phenotypic submodels), followed by a final submodel that predicted yield (dry weight submodel). Once trained the compound models followed a common principle of operation for generating predictions, illustrated on Figure 4.8.

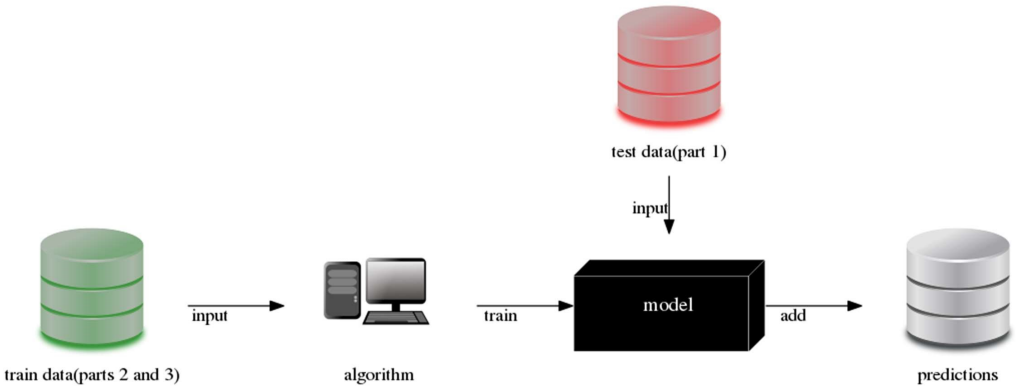

Figure 4.7 Training and testing for a single cross-validation fold. In this example k=1*, which meant that the training data came from parts 2 and 3 (years 2015 and 2016) and the testing data was part 1 (year 2011). The predictions were then stored together with predictions from the rest of the folds.*

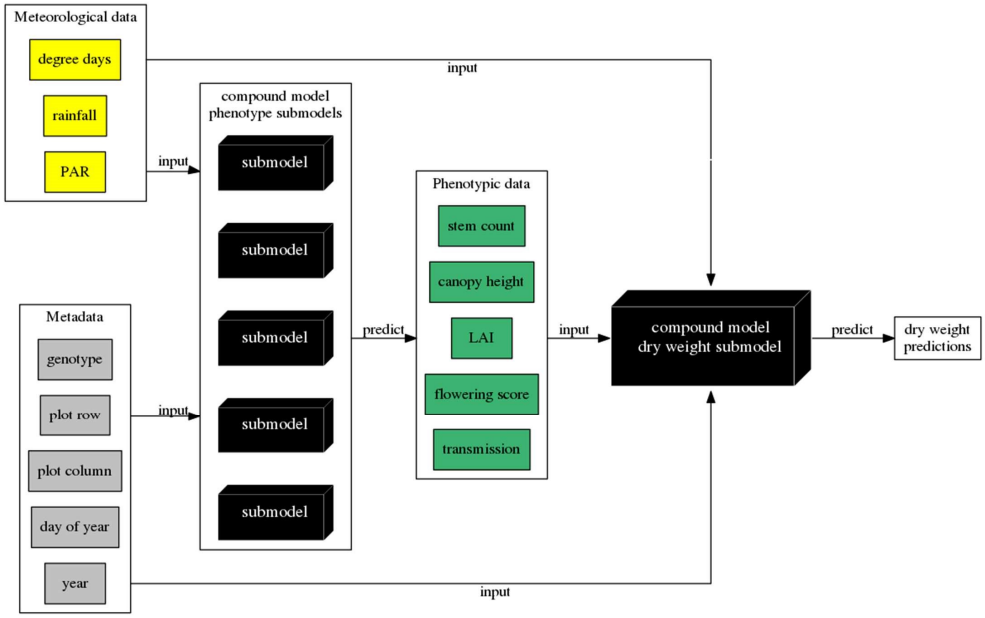

Figure 4.8 Principle of operation of the compound model - metadata and meteorological data served as input to the model. A number of phenotype submodels predicted each phenotypic variable, using the input data and in some cases other predicted phenotypic variables. Finally, all the input and predicted phenotypic data was passed to the dry weight submodel, which outputted the dry weight prediction. The meteorological data was the cumulative daily values since emergence – this allowed the model to simulate phenotypic and dry weight values for each day of the growing season.

Training a compound model meant training each phenotypic submodel in turn to predict its target variable, by providing real meteorological, metadata and phenotypic data to the model training procedure. Afterwards the same approach was applied to training the dry weight submodel, using all phenotypic, metadata, meteorological and yield data available.

A major advantage of compound models over simple machine learning models was that for generating predictions, the compound models accepted only meteorological and metadata as input, just like mechanistic models like MISCANMOD and BPM. While the exact structure of the models in this category varied, it followed the same principle: each submodel predicted a phenotypic variable, after being provided with meteorological and metadata, and in some cases phenotypic data predictions as input. The submodels were simple machine learning models, trained using one of the algorithms listed under section 4.2.3 (linear regression, ANN, random forests, etc.) to take a set of variables as input (meteorological, metadata, and in some cases phenotypic data) and predict another phenotypic variable or yield. When predicting, each submodel was used in turn to predict a phenotypic variable, and afterwards all meteorological data, metadata and predicted phenotypic data was provided to the final submodel which produced a prediction for yield.

This subsection reviews two different approaches, which were used to create the two models of this category: the naïve model and the genetic algorithm (GA) model. The two models were tested against ABR61 data from the year 2016 and were trained on the rest of the ABR61 dataset (years 2011 – 2015). Their modelling accuracies were compared with APBPM which was also trained and tested on the same data. The testing procedure is described in the following subsections.

Naïve model

The naïve model was developed following a naïve approach, which assumed that no phenotypic variables were independent of each other, so no phenotypic variable could be predicted from another. Once trained, the model made predictions following the structure illustrated in Figure 4.9.

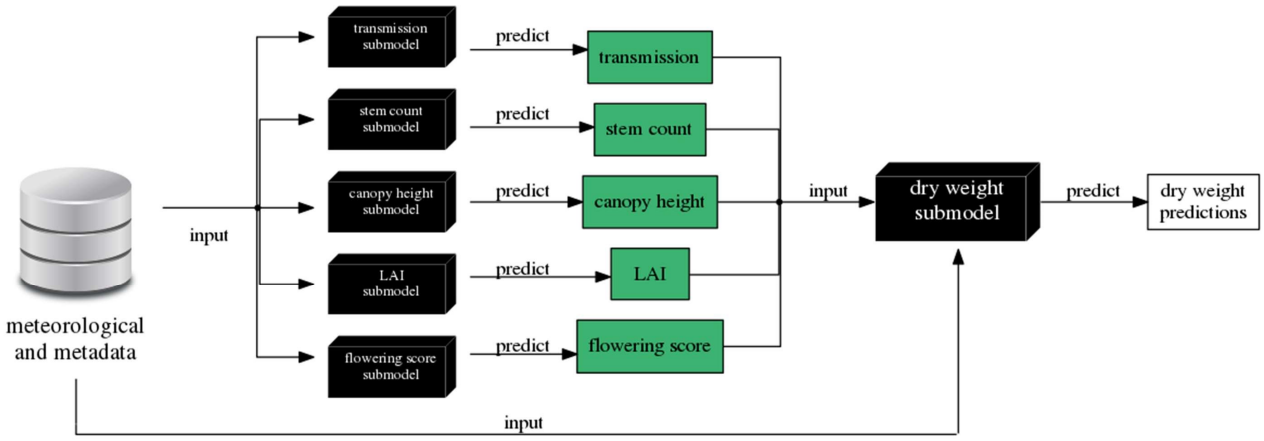

Figure 4.9 Principle of operation of the naïve compound model – meteorological and metadata are used as input for each phenotypic submodel. The submodels predict each phenotypic variable and the predicted phenotypic variables and the meteorological and metadata are passed to the dry weight submodel which predicts yield.

Each phenotypic variable submodel used all input variables to the model (metadata and meteorological variables) and none of the other predicted variables, to predict their corresponding phenotypic variable. The dry weight submodel took all input and predicted variables, and produced yield predictions.

As mentioned before, to train the model, each submodel needed to be trained. The training data for each phenotypic submodel consisted of the meteorological and metadata variables, as well as the target phenotypic variable of that submodel. The training data for the dry weight submodel was all available variables (phenotypic, meteorological and metadata) as well as the dry weight data. This data was comprised of real measurements. Predicted phenotypic data was not used for training other submodels.

However, before the submodel could be trained, the algorithm for training each submodel needed to be chosen. For each submodel there was a choice between using one of the five machine learning algorithms listed under 4.2.3. Each machine learning algorithm produced a candidate submodel. The accuracy of the candidate submodels was determined using a cross-validation procedure. The machine learning algorithm that produced the most accurate candidate submodel, was chosen to produce the submodel. This process is referred to as submodel choice and is described in more detail later in this section. This process was not automated, the model comparison was an experiment that was run separately, and its results were used to choose the submodels.

As the naïve compound model was used to determine importance of phenotypic data and measurement time, which is described in the following chapter, it was infeasible to use the “train” function from the “caret” package to train each submodel. The execution time of the function was long because of the overhead caused by the parameter tuning procedure, which trained each submodel with multiple sets of parameter values to find the ones that led to the best performance. Thus, for training most submodels the “caret” procedure was substituted with the direct use of the “underlying functions” from Table 4.2.

The “train” function was still used for two of the machine learning algorithms: ANN and SVM, because their parameter values were critical for their performance (Chapelle *et al.*, 2002; Leung *et al.*, 2003). For the rest of the machine learning methods, if any default parameter values were specified by the corresponding function, they were used during training.

The linear regression submodels were trained with an intercept term. In GBM the n.trees parameter was set to 1000 and the rest of the parameters (interaction.depth, shrinkage, n.minobsinnode) were left at their default values. All parameters for k-NN and random forests were left at their default values as well.

As mentioned before, for each phenotypic submodel, cross-validation was used to find the algorithm that produced the candidate submodel with the most predictive power. It was performed on the training part of the ABR61 dataset (years 2011 – 2015). The data was divided into four parts, one for each year, as shown on Table 4.4. The cross-validation procedure was identical to the one used for the simple machine learning models, with the exception that the years 2012 and 2014 could be used as they contained the necessary phenotypic measurements. No data from 2016 were used during cross-validation, as it was set aside for testing purposes. Using the predictions from the cross-validation, RMSE was calculated for each candidate submodel against each variable. The candidate submodels were grouped by the variable they predicted and the ones with the smallest RMSE from each group were chosen to be the submodels for the naïve model. An example of the first fold of cross-validation, done for choosing the best machine learning algorithm for the canopy height submodel is illustrated in Figure 4.10.

CV part	2011	2012	2014	2015
k	1	2	3	4

Table 4.4 Years from the dataset in each cross-validation fold.

The procedure for picking an algorithm for the dry weight submodel was similar to the one described above. However, due to lack of yield measurements during 2012 and 2014, cross-validation folds could only be formed from two years – 2011 and 2015.

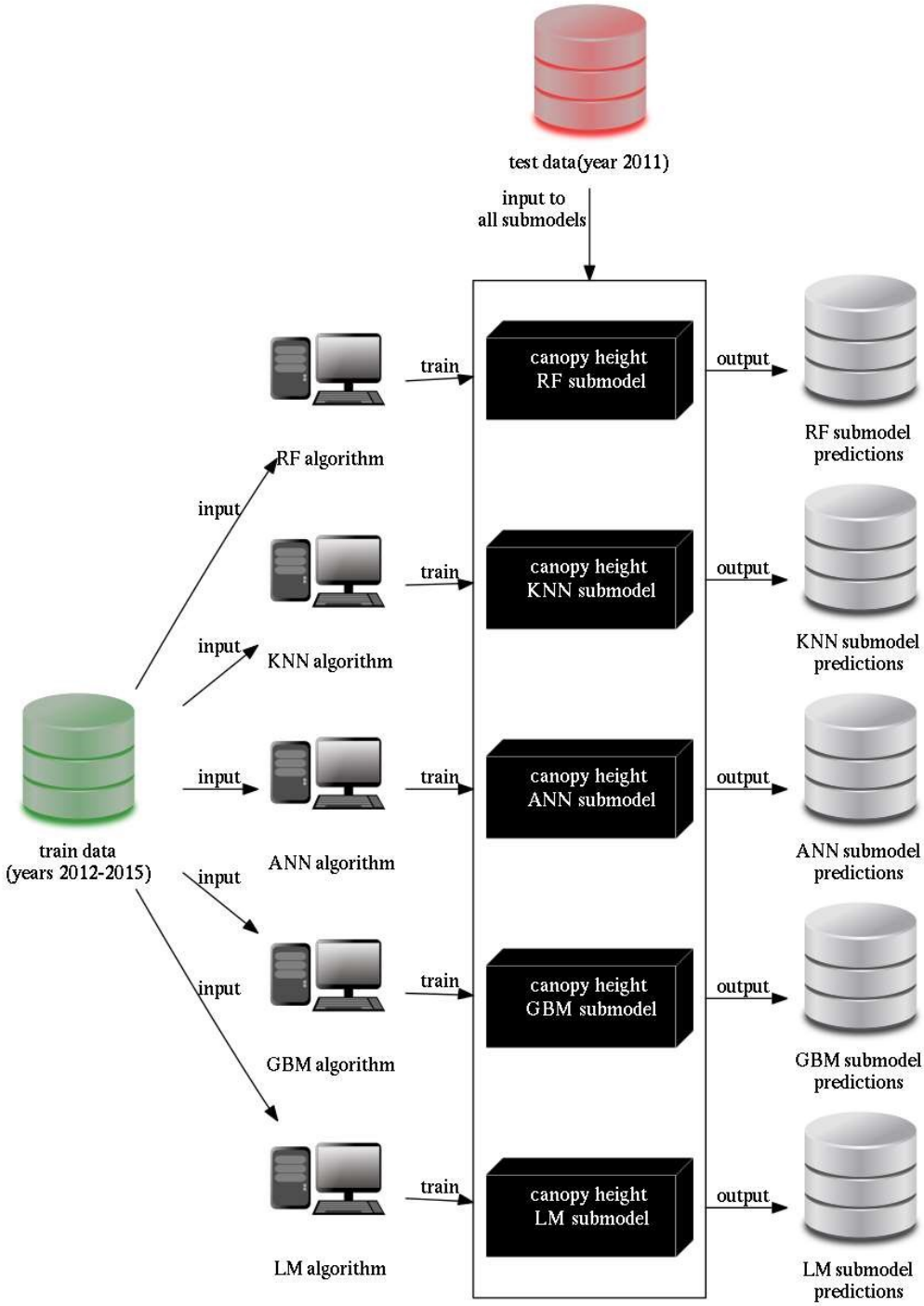

Figure 4.10 Example of the first cross-validation fold done for choosing the machine learning algorithm for building the canopy height submodel. The training data (ABR61 years 2012-2015) was provided to each machine learning algorithm which trained a corresponding machine learning submodel. Each of these submodels was tested by generating predictions over the unseen test data, which for the first cross-validation fold was the ABR61 year 2011 dataset. The predictions produced from each individual submodel were pooled together with predictions produced from the same submodels in the other folds. These were finally used to calculate each submodel's RMSE.

Cross-validation was performed with the two folds, RMSE was calculated for the candidate submodel produced by each algorithm, and the one with the smallest error was used as the dry weight submodel. After the choice for the submodels was made, each submodel was retrained over the whole training dataset. The naïve model was then used to generate yield predictions over the unseen 2016 test data, and RMSE and R^2 were calculated over these predictions.

GA model

Genetic algorithm (GA) models, as referred to in this chapter and chapter 5, denotes models that were produced using the genetic algorithm method described in this section. The structure of the GA model was generated using genetic algorithms (GA), that explored a range of possible model structures and identified the ones that worked best for building accurate models. Each GA model structure solution had to be represented in a GA chromosome format, which allowed the GA to mutate and cross solutions. A chromosome was a list of values that defined which machine learning algorithms would be used for each submodel (e.g. flowering score submodel), as well as the input variables to these submodels. An example chromosome is shown in Table 4.5. Each chromosome was made of six genes, presented as columns in the table. The variable in each column was either the phenotypic variable or dry weight, which the corresponding submodel would predict. The algorithm was the machine learning method used for building the submodel (RF, SVM, etc.). Finally, position defined the position of the submodel in the graph, which determined its input variables. The example model structure from Table 4.5 is illustrated on Figure 4.11.

Variable	Flowering score	Stem count	Canopy height	Transmission	LAI	Dry weight
Algorithm	ANN	LM	ANN	SVM	RF	GBM
Position	1	3	2	2	3	4

Table 4.5 Example GA chromosome which describes the model structure and algorithm choice for each submodel.

For making predictions, the input data for the GA model was, as was the case for the naïve model, the meteorological data and metadata (which will be referred to as the

model input data in this chapter). Flowering score is on position 1 which means that its submodel would only receive the model input data as input. As defined in Table 4.5 an ANN would be used for the submodel. The next position has two submodels: canopy height (ANN) and transmission (SVM). Their input data will be made up of the model input variables, as well as the variable from the previous position – the predicted flowering score. The same principle is applied for the submodels at the following position (stem count (LM) and LAI (RF)), making their input the model input variables, predicted flowering score, predicted canopy height and predicted transmission. Finally, the dry weight submodel (GBM) takes the model input data and all the predicted variables from the previous positions and predicts yield.

At each generation of the GA, there were multiple solutions. Each solution contained a chromosome, which defined a GA model. Each GA model was trained and cross-validated to get its accuracy. The aim of GA was to modify the chromosome of each solution, to get more accurate models. It could do that by changing the machine learning algorithm used for each submodel, or by changing the submodel position in the compound model. GA could also combine or "cross" two solutions, to produce a better solution. This is described in more detail later in this section. The solutions that produced the best GA models were kept in the next generation. This process was repeated until the algorithm has found an optimum solution, which could not be improved any more.

GA could generate solutions by assigning random values for each submodel and position. Submodel values were any of the five machine learning methods, and position values were numbers between 0 and 5, where numbers 1-5 denoted the location of the submodel in the GA model and 0 instructed the submodel to be removed from the GA model. While a submodel could be dropped altogether, two submodels predicting the same variable may not appear in the same GA model. GA could also cross two solutions, by picking a breakpoint column randomly, getting all the columns up to that breakpoint from the first solution, and adding to that the columns after the breakpoint from the second solution. Finally, individual genes could be mutated by randomly picking new values for the submodel and position.

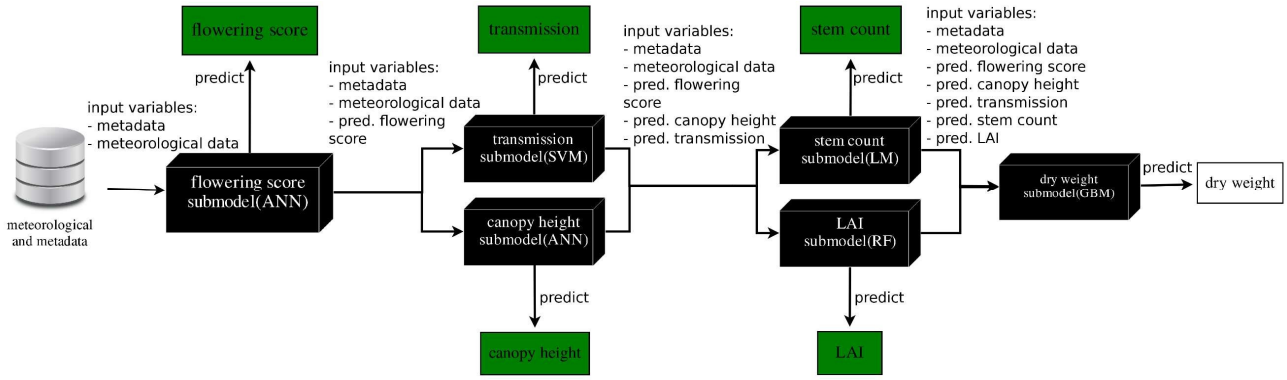

Figure 4.11 Example GA solution for the GA model structure and submodel choice. The chromosome that defines the solution in the diagram is shown in Table 4.5. The input for the submodels from each stage is the model input data (metadata and meteorological data), as well as the predicted variables from the previous stages. The position number defined where each submodel was in the GA model structure. For a given submodel, any submodels with smaller position numbers appeared before that submodel, submodels with the same number were in the same stage, and submodels with a greater position number appeared after that submodel.

The genetic algorithm used for building GA models is summarised in Figure 4.12. A few parameters influenced the exact actions of the algorithm. These were:

- *pop_size* – governed the population size, i.e. the number of solutions in each generation. This was set to 100.
- *keep_rate* – the percentage of solutions from each generation that would be advanced to the next generation. The value used was 20%, which meant that 10 of the best solution from each generation were kept for the next one.
- *cross_rate* – percentage of each new generation that was filled with solutions generated by crossing two solutions together. This was set to 60%.
- *migration_rate* – percentage of each new generation that was filled with randomly generated solutions. This was 20%.
- *mutation_rate* – the probability that a single gene from a solution generated by crossing would mutate. It was set to 20%.

The algorithm started with randomly generating *pop_size* number of solutions which formed the first generation. The accuracy of each GA model was assessed using cross-validation. Data from the years 2011 and 2015 from the ABR61 dataset was used for running two cross-validation folds, one where models were trained on the 2011 data and predicted 2015 data and one vice versa.

Training each GA model resembled the procedure used for the naïve model. Only real phenotypic, metadata and meteorological data was used to train each submodel. For phenotypic submodels in the first stage, the training data consisted of the target phenotypic variable, as well as meteorological and metadata variables. For submodels from subsequent stages, the training data from the previous stage was used and the target phenotypic variable was added. For the example shown in Figure 4.11, the flowering score submodel would receive the real meteorological, metadata and flowering score measurements as training data. The transmission submodel from the next stage would receive the same training dataset as the flowering score submodel, as well as real transmission data.

Algorithm parameters:

- *pop_size* = 100
- *keep_rate* = 20%
- *cross_rate* = 60%
- *migration_rate* = 20%
- *mutation_rate = 20%*

Generate *pop_size* solutions
Add solutions to current generation

Repeat:
Calculate RMSE for each solution from current generation
Sort solutions in generation in ascending order by RMSE (best solutions were first)

Get top *keep_rate* solutions from the current generation
Add solutions to new generation

Repeat:
Randomly choose two solutions from population (using CDF)
Cross solutions
Mutate genes of new solution
Add solution to new generation
until *cross_rate* of *pop_size* number of solutions are generated

Generate *migration_rate* of *pop_size* solutions randomly
Add solutions to new generation

current generation = new generation

if no change last 20 generations:
increase *migration_rate* and *mutation_rate*
decrease *keep_rate* and *cross_rate*

if changes last 20 generations:
decrease *migration_rate* and *mutation_rate*
increase *keep_rate* and *cross_rate*

until stopped

Figure 4.12 GA algorithm for finding the optimal GA model structure and choosing the submodels.

On the first cross-validation fold, each GA model was trained against the 2011-2014 ABR61 data, and generated predictions on the 2015 ABR61 dataset. The equivalent procedure was used on the second fold, where the training data consisted of years 2012-2015 and test data was year 2011. The accuracy measurement was RMSE, which was calculated using the yield predictions generated by the GA models and the real data.

The algorithm then took three steps for generating the next generation of solutions. A percentage of the top (or elite) solutions from this generation, defined by the *keep_rate* parameter, was kept for the next generation. The elite solutions were the ones with the smallest RMSE.

The algorithm randomly chose solutions from the current generation that it would cross together and make new solutions. The probability of choosing solution x_n was determined by a cumulative probability function P_c. The function was chosen so the first solutions had the highest chance of being chosen, and the probability decreased the further a given solution was in the list of solutions. Thus, the function assigned the greatest probability to the best solutions and probability was smaller the worse a solution was. The cumulative probability function (P_c) was:

$$\mathrm{P}_c(\mathrm{x}_\mathrm{n}) = \mathrm{P}_c(\mathrm{x}_{\mathrm{n}-1}) + \frac{\mathrm{P}_c(\mathrm{x}_{\mathrm{n}-1})}{\mathrm{d}}$$

Where:

- d was a divisor parameter which varied depending on the population size. For $\mathrm{pop\_size} = 100$, $\mathrm{d} = 1.05228$. The value of this parameter was chosen to make the probability of the cumulative probability of the last solution $\mathrm{P}_c(\mathrm{x}_{100}) = 1.0$. This ensured that each of the 100 solutions in the population could be picked using the cumulative probability, because the algorithm picked a solution by generating a random number between 0 and 1. This is explained in more detail later in this section.
- The probability of picking the first solution x_1 was defined manually and was set to be 0.05. This value was picked by visually inspecting a plot of the individual probability for each solution (Figure 4.13).

The algorithm sorted the solutions in the generation by RMSE, with the best solutions put in the first places. The individual probability P for each solution was calculated by taking the difference between its cumulative probability and the cumulative probability of the previous one:

$$P(\mathrm{x_n}) = P_c(\mathrm{x_n}) - P_c(\mathrm{x_{n-1}})$$

The probability for each of the 100 solutions is shown in Figure 4.13. Thus, the distribution gave higher probabilities to the top solutions as they produced the most accurate models. The probability declined as n increased, but it never became 0. The probability for x_{100} was 3.22×10^{-4}.

The algorithm used a random number generator that generated real numbers between 0 and 1, to choose the solutions for crossing. After generating a number, it picked the first solution with cumulative probability higher than the chosen number. After picking two solutions from the generation in this manner, they were crossed in the way described earlier in this subsection. The genes of the new solution were mutated with probability determined by the *mutation_rate* parameter. Finally, the newly generated solution was added to the new generation. The choose/cross/mutate process was repeated until a percentage (determined by *keep_rate*) of the new generation was generated this way. Only solutions created by

crossing were mutated – no mutation was applied to the elite solutions, as this could have caused their RMSE to increase.

The final way of filling in the new generation was to generate new individuals randomly. The number of individuals generated this way was determined by the *migration_rate* parameter.

The three parameters *keep_rate*, *cross_rate*, *migration_rate*, and *mutation_rate* presented a means of controlling whether the algorithm prioritised keeping good solutions and trying to generate better ones by crossing them, or exploring different regions of the solution space, and hoping to discover better solutions. The parameters were allowed to change when no better solutions had been found, because it was assumed that the algorithm might be stuck in a local minimum – a solution which was the best among its neighbouring solutions, but which compared poorly to the best global solution. If the best solution from the previous 20 generations did not change, *keep_rate* and *cross_rate* were decreased and *migration_rate* and *mutation_rate* were increased, allowing for more exploration. When changes appeared in the last 20 generations, *keep_rate* and *cross_rate* increased and *migration_rate* and *mutation_rate* decreased, to take advantage of the newly discovered region.

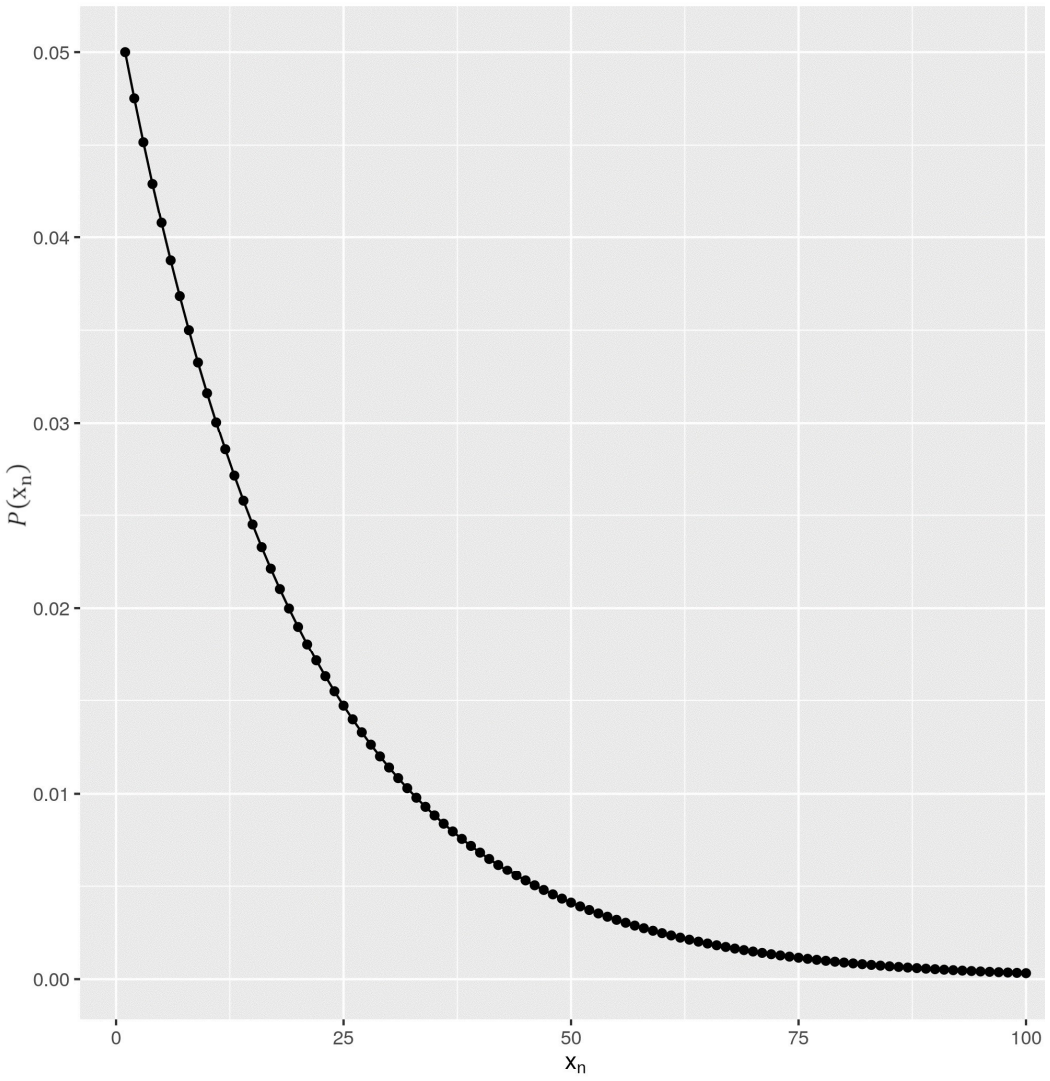

Figure 4.13 The probability for picking each solution from GA generation x_n for crossing. The population size was 100, d was 1.05228, and $P(x_1)$ was 0.05.

A flowchart of a single generation of the GA algorithm is illustrated on Figure 4.14, using the default starting values for keep_rate, cross_rate and migration_rate. The GA was run until the best solution in the generation had not changed in the last 1000 generations. It was then decided that it could not find a better solution. The best solution from the last generation was used to build the final GA model. This model was trained on the ABR61 dataset, years 2011 – 2015 and was tested against the unseen 2016 ABR61 data. RMSE and R^2 were calculated using the predictions and actual dry weight values from the year 2016. This allowed a comparison of the modelling accuracies of the GA model, the naïve model and APBPM.

4.2.5 Software

The implementations of the individual machine learning algorithms, used to train simple machine learning models and the submodels of the compound models, were provided by the various R packages (R Core Team, 2017) mentioned in section 4.2.3. The procedure for creating bagged models was implemented in R. All other methods, including cross-validation, error calculation, the naïve and GA models, along with their structure and submodel choice procedures were implemented in the Python 2.7 programming language (Python Software Foundation, 2013).

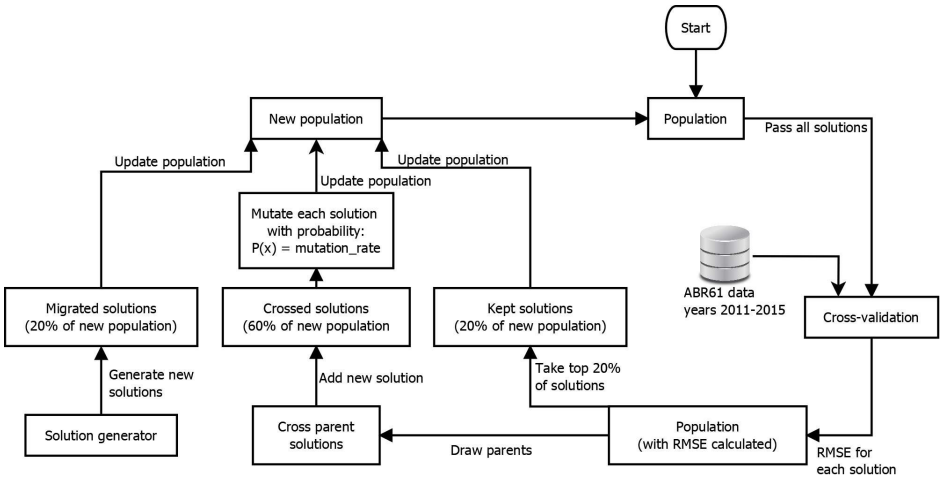

Figure 4.14 Flowchart of a single GA generation. The process starts with the population which is passed to the cross-validation procedure, which evaluates the RMSE for each GA model (referred to as solution in the graph) in the population. 20% of the best solutions are kept for the next population. 60% of the new population is created through crossing individuals from the population. The individuals used for crossing (parents) are picked randomly each time, using the cumulative probability function, which gave the best solutions better chance of being picked. Each gene from the crossed solutions was mutated with a probability defined by the mutation_rate parameter. The final 20% of the new population was generated by a random solutions generator. The kept, crossed and newly generated solutions were all added to the new population. Finally, the new population replaced the old population and the loop repeated. The percentages for kept, crossed and new solutions were determined by the GA algorithm variables keep_rate, cross_rate and migration_rate, and could change if the algorithm failed to find better solutions over time. The percentages in the diagram and this caption are just example values.

4.3 Results

4.3.1 Simple machine learning models

Model accuracy comparison

The results of the cross-validation and calculation of RMSE and R^2 are shown in Figure 4.15 and 4.16 respectively. Table 4.6 contains dry weight descriptive statistics for the whole ABR61 dataset, to help with understanding the implication of the RMSE values for each model. The two measures of accuracy show random forests as the most precise model closely followed by GBM and k-NN. For that reason, these three models were selected to be used in the work described in the following chapter. It was assumed that the most accurate models made the best use of the training data as they "learned" the most from it. Thus, they were the best indicator of how much "knowledge" there was in it.

Most models were at least as accurate as APBPM, with exception of the bagged version of SVM, ANN and the two LM models. While the best three models present a great reduction in error from APBPM, it cannot be attributed only to internal differences between the models. The training data provided to the machine learning models for prediction contained phenotypic variables, which APBPM did not use for driving its model. As mentioned previously, the use of phenotypic variables for predicting yield was a weakness of the machine learning models, because they were impractical in a real scenario, where yield would need to be predicted from meteorological data only. However, the approach produced more accurate models, which would be more useful for finding the importance of variables and measurement times for building predictive models.

Another difference in the training data between APBPM and the machine learning models was that APBPM could use data points, where dry weight or other measurements were missing, whereas machine learning models omitted them from the training data. Thus, differences between APBPM and the machine learning models were due to several reasons which could not be disentangled with a simple comparison of RMSE and R^2.

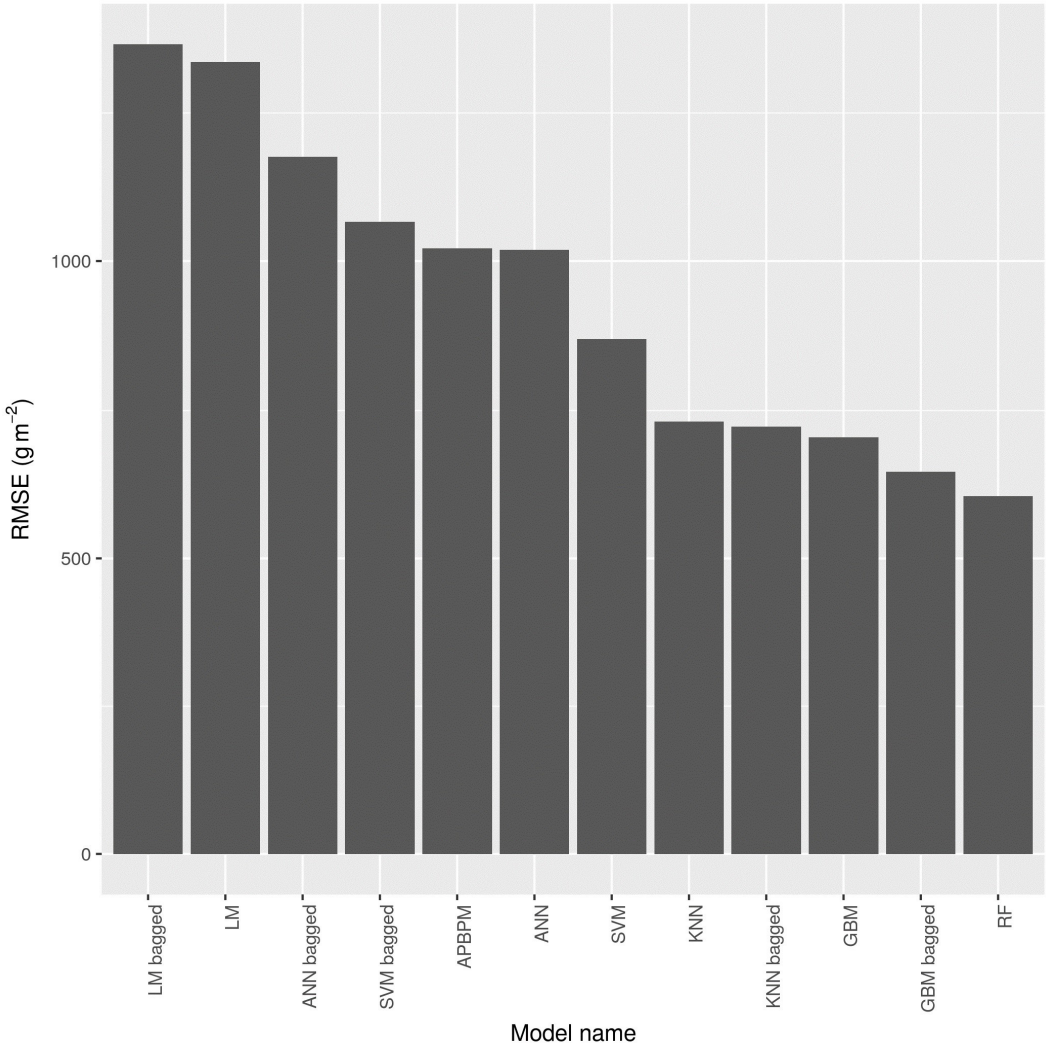

Figure 4.15 Comparison of simple machine learning models' and APBPM's RMSE values (smaller values mean more accurate models). RMSE was calculated by cross-validation over the ABR61 dataset. Table 4.6 shows descriptive statistics for dry weight to put the RMSE values for each model in perspective.

Genotype	Mean	Standard deviation	Minimum value	25%	50%	75%	Maximum value
EMI-11	1269	861	17	686	981	1956	3209
Giganteus	1516	1278	3	624	1262	2198	5798
Goliath	1299	943	6	468	1226	1812	3881
Sac-5	1496	1117	1	768	1149	2217	4527

Table 4.6 Descriptive statistics for the dry weight values in the whole ABR61 dataset. The percentile columns denote the first quartile (25%), the second quartile (median, 50%) and the third quartile (75%).

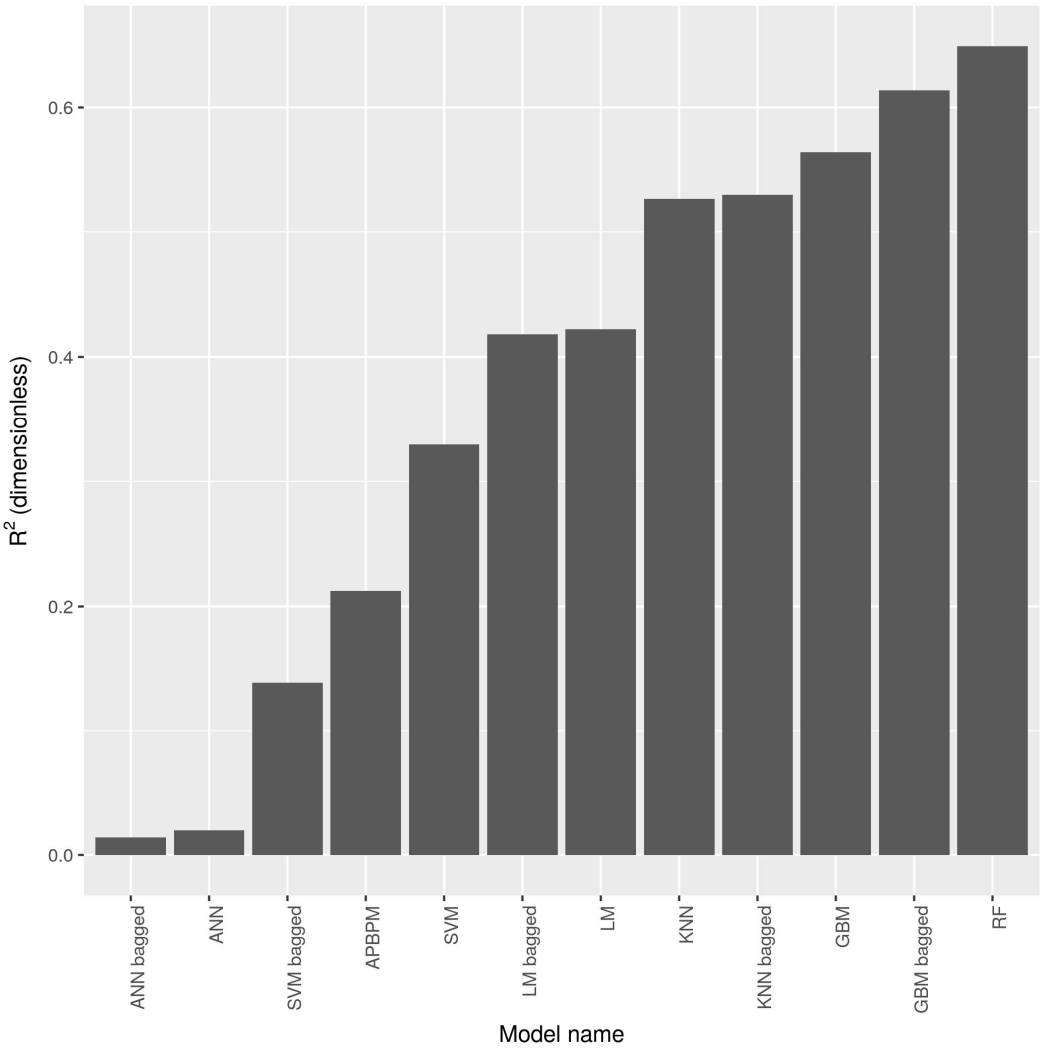

Figure 4.16 Comparison of simple machine learning models' and APBPM's R^2 values (large values meant that the model's predictions correlated more with the actuals, so the model was more accurate). R^2 values were calculated by cross-validation over the ABR61 dataset.

Comparison of standard and bagged model versions

Shapiro-Wilk test of normality showed that RSE values for all models were not drawn from normal distributions. The p values from each test results are shown in Table 4.7.

Method	p value
KNN	9.85×10^{-15}
KNN bagged	4.48×10^{-14}
ANN	8.15×10^{-10}
ANN bagged	2.92×10^{-16}
LM	1.40×10^{-19}
LM bagged	6.04×10^{-20}
SVM	1.01×10^{-11}
SVM bagged	3.02×10^{-12}
GBM	5.26×10^{-13}
GBM bagged	7.10×10^{-12}

Table 4.7 P values from the Shapiro-Wilk test, which tested normality of the RSE distribution for each model. Due to the low p values, the null hypothesis that RSE values came from a normal distribution was rejected.

This meant that a t-test could not be used for testing for difference in the means of the distributions from which the errors (root squared error (RSE) value) of the standard and the bagged version of each model were drawn. The smaller p value from the two one-tailed Wilcoxon signed-rank tests is shown in Table 4.7. Depending on which model the p value belonged to, the bagging effect was assigned as either "Increased error" or "Decreased error" in the table, indicating whether the bagged model increased or reduced the mean of the error distribution. Thus, the p values signify the statistical significance of the effect of using each bagged model version. The bagged versions of only two models decreased the mean of the error distribution, and for one of them (KNN) the RMSE difference was only ~7.9g dry weight. These were also the two models that did almost as well as random forests. The bagging effect on other three models ANN, LM, and SVM was detrimental – the bagged version of the model increased the error in those three cases. This could be because bagging is most efficient with weak learners (Breiman, 1998, 2001a), SVM, LM and ANN are all strong learners, whereas the base learners of GBM are weak, and the predictions of KNN could easily be varied by perturbations in the training dataset. Bagged versions of KNN and GBM had relatively small gains in modelling accuracy,

which did not outweigh the overhead of training 100 base learners, compared to just training one with the standard model.

Method	Bagging effect	p value	Error difference: $RMSE_{standard} - RMSE_{bagged}$
KNN	Reduced error	0.02	7.981
ANN	Increased error	0.01	-157.02
LM	Increased error	9.7×10^{-6}	-29.931
SVM	Increased error	3.57×10^{-5}	-196.707
GBM	Reduced error	2.35×10^{-12}	57.605

Table 4.8 Results of one-tailed Wilcoxon signed-rank test between standard models and their bagged versions. The test was used to establish if there was a difference between the distributions of error (RSE) for each pair of models. The bagging effect denotes whether the bagged version had increased or reduced RMSE value compared to that of the standard model. The p value came from the result of the Wilcoxon signed-rank test and denotes statistical significance.

4.3.2 Compound models

Naïve model

The results from the cross-validation procedure for choosing machine learning algorithms for each submodel (e.g. canopy height submodel, stem count submodel) in the naïve model are shown on Figure 4.17 (descriptive statistics are shown in Table 4.6 for dry weight and 4.9 for all other variables). For each variable (e.g. canopy height), the algorithm that produced the submodel with the smallest RMSE was chosen to be the algorithm training the variable's submodel. For canopy height, flowering score and transmission, this was random forests. Linear regression (LM) outperformed the other candidate submodels for dry weight and stem count and it was used for predicting these variables. For LAI, the best performing candidate submodel was SVM, with $\mathrm{RMSE} \approx 2.61\ (\mathrm{dimensionless})$. It was followed closely by random forests with $\mathrm{RMSE} \approx 2.71\ (\mathrm{dimensionless})$. It was noted that SVM required considerably more time for training to complete, because "caret" was used to tune its parameters.

To establish the exact difference, a RF and SVM model were trained on the training dataset to predict LAI. This was repeated 10 times, and the time was measured. RF took on average 4.75 seconds per training, whereas SVM needed 59 seconds. As the models developed in this chapter were continuously retrained as part of the work described in the following chapter, the improvement in RMSE of 0.1 that SVM offered did not outweigh the overhead of significantly longer training time. Thus, RF was used for the LAI submodel. The final naïve model with the submodel choices is shown on Figure 4.18.

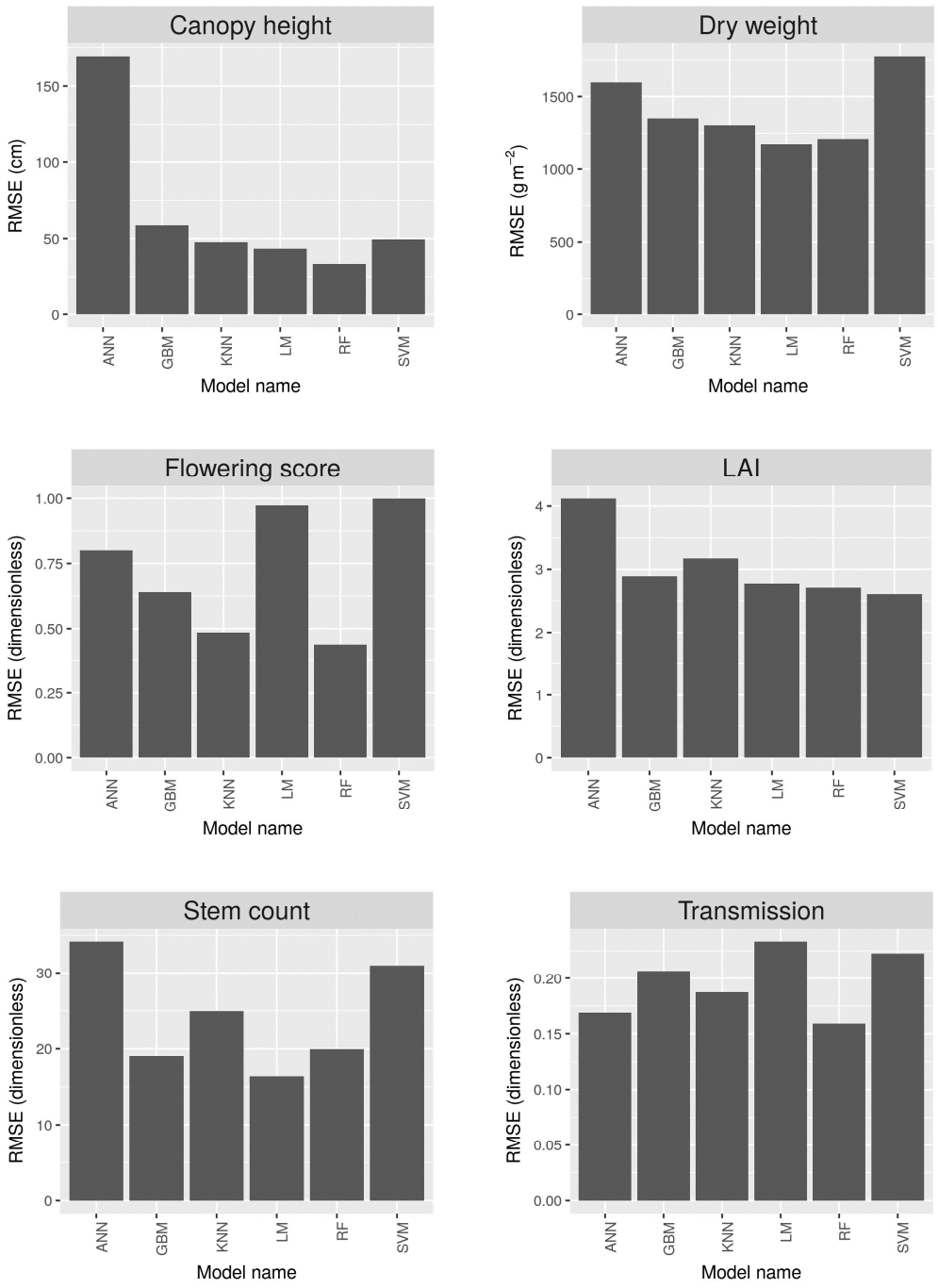

Figure 4.17 Comparison of candidate submodels generated by each machine learning algorithm for each variable using cross-validation for each module of the naïve model. Training individual models for all combinations of algorithm and variable, allowed to identify whether different machine learning algorithms were better at training models to predict a particular variable well. The submodels with the smallest RMSE were chosen to predict the corresponding variable, with the exception for LAI, where RF was preferred to SVM, because of its faster training time. Descriptive statistics for dry weight can be found in Table 4.6. For all other variables presented in the figure, these statistics are shown in Table 4.9. There could have been models which were better for predicting a given phenotypic variable for one genotype but not another – this was not tested, as it was assumed that a model would work for each genotype equally well.

Variable	Genotype	Mean	Standard deviation	Minimum value	25%	50%	75%	Maximum value
Canopy height	EMI-11	110.72	43.62	16	74	121.75	149	188
Canopy height	Giganteus	191.3	83.72	10	114.5	212.5	263	321
Canopy height	Goliath	131.3	49.35	3	93.12	138	174.88	265
Canopy height	Sac-5	191.42	83.35	3	124	208	264	360
Flowering score	EMI-11	0.89	1.3	0	0	0	1	5
Flowering score	Giganteus	0	0	0	0	0	0	0
Flowering score	Goliath	0.29	0.65	0	0	0	0	3
Flowering score	Sac-5	0	0	0	0	0	0	0
LAI	EMI-11	3.54	2.77	0	1.27	3.07	5.44	16.91
LAI	Giganteus	4.38	3.7	0	1.77	4.01	6.11	28.29
LAI	Goliath	3.89	3.6	0	1.2	2.87	5.71	18.38
LAI	Sac-5	2.97	2.16	0	1.14	2.72	4.4	11.17
Stem count	EMI-11	47.68	24.79	8	29	47.33	62	153
Stem count	Giganteus	21.62	10.03	1	16.82	21	24.83	86
Stem count	Goliath	31.76	17.31	1	21.16	27	35.66	105
Stem count	Sac-5	14.4	6.98	1	8.72	12.42	20	39
Transmission	EMI-11	0.18	0.25	0	0.01	0.05	0.27	0.93
Transmission	Giganteus	0.19	0.25	0.01	0.03	0.07	0.27	1
Transmission	Goliath	0.22	0.26	0	0.02	0.11	0.34	0.99
Transmission	Sac-5	0.28	0.24	0.01	0.09	0.19	0.39	0.96

Table 4.9 Descriptive statistics for the canopy height, flowering score, LAI, stem count and transmission in the whole ABR61 dataset.

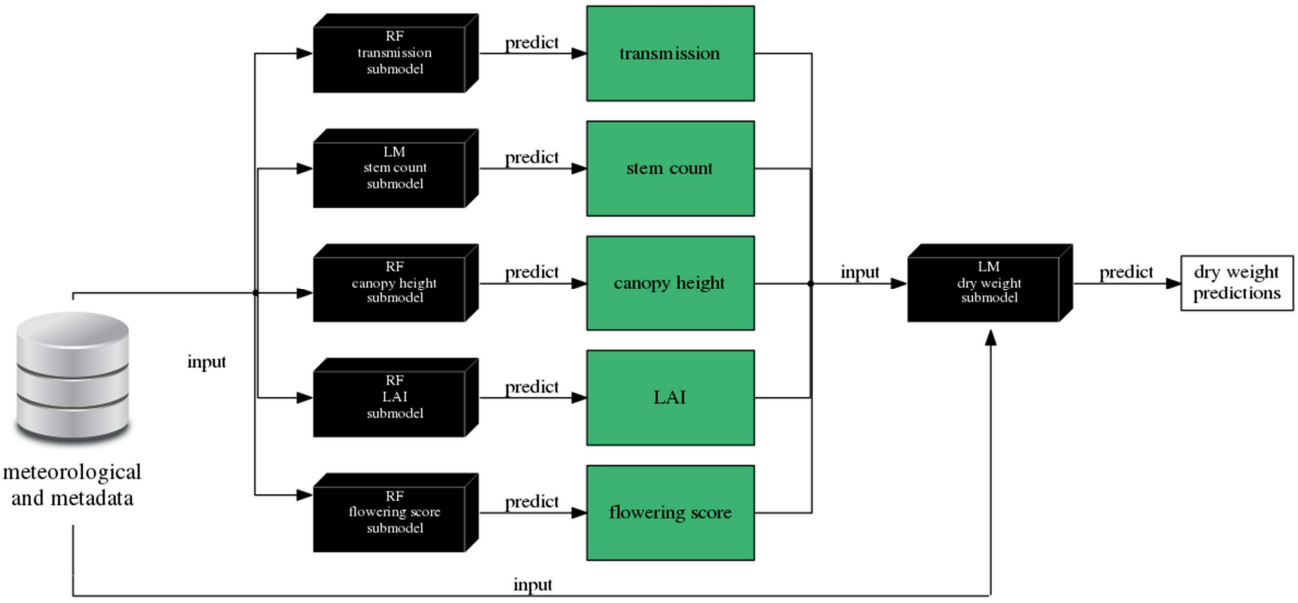

Figure 4.18 Naïve model structure and submodel choice – RF was chosen for the transmission, canopy height, LAI and flowering score submodels, and LM was used for the stem count and dry weight submodels.

GA model

The GA was run multiple times, to test the GA implementation and then the GA was run a final time and reached its best solution in the 20th generation. The RMSE of the best solutions from each generation up to generation 30 are shown in Figure 4.19. It was calculated through cross-validation on the ABR61 data, years 2011 and 2015, using predicted and actual dry weight values. The GA was run sufficiently long to make sure it could not find a better solution. The algorithm reached 1313 generations but it failed to find any better solutions beyond generation 20.

Figure 4.20 shows the change in the GA parameters: keep rate, cross rate, migration rate and mutation rate. The parameters started changing at generation 40, as the GA had not encountered any change in the best solution for the previous 20 generation. As no better models were found in the following generations, the keep and cross rate kept decreasing to their minimum of 11% and 21% respectively, and the migration and mutation rate kept increasing up to their maximum of 68% and 40%. This allowed the algorithm to explore random regions of the solution space while still retaining the best solutions it had found. As no better solution was found in the following generations, the parameters retained these values until the algorithm was stopped at generation 1313. The structure and submodels of the best solution, found in the 20th generation are shown on Figure 4.21.

The GA model has dropped the canopy height variable, so it did not train a canopy height submodel, nor did it use canopy height for training or for making dry weight predictions. Most likely the accuracy of the model shown in Figure 4.21 would have suffered if a canopy height submodel was added. This could not be established because the implementation of GA did not retain any records of models from previous generations, apart from their RMSE, so the exact decisions taken by the GA could not be observed in retrospect.

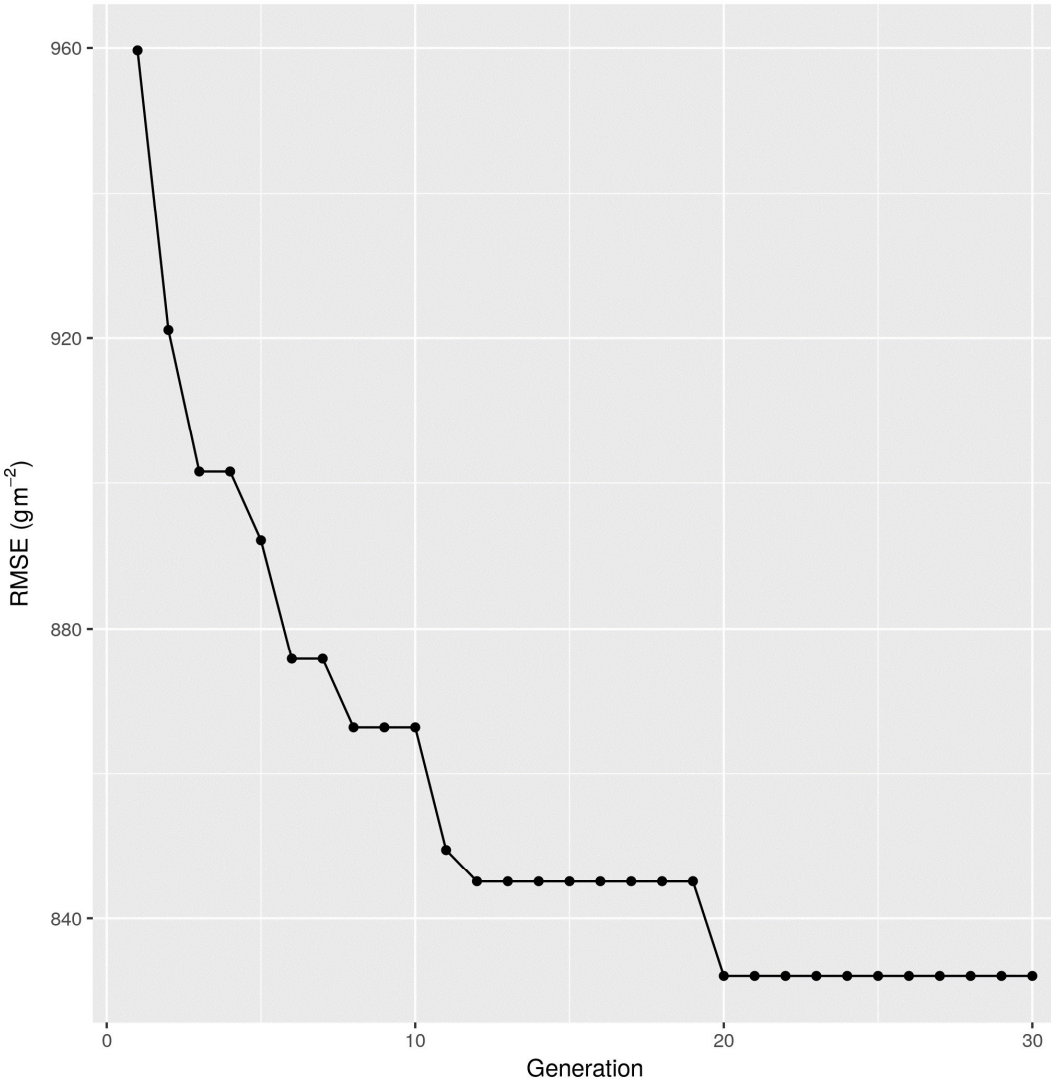

Figure 4.19 RMSE of the best model in each generation created by the genetic algorithm. RMSE was calculated through cross-validating over the ABR61 dataset, years 2011 and 2015, using the predicted and actual dry weight values. Starting from the 20th generation, the accuracy of the best model was $RMSE \approx 832$. GA was left to run up to generation 1313, but it failed to find any better models. Note that y axis in this plot is truncated.

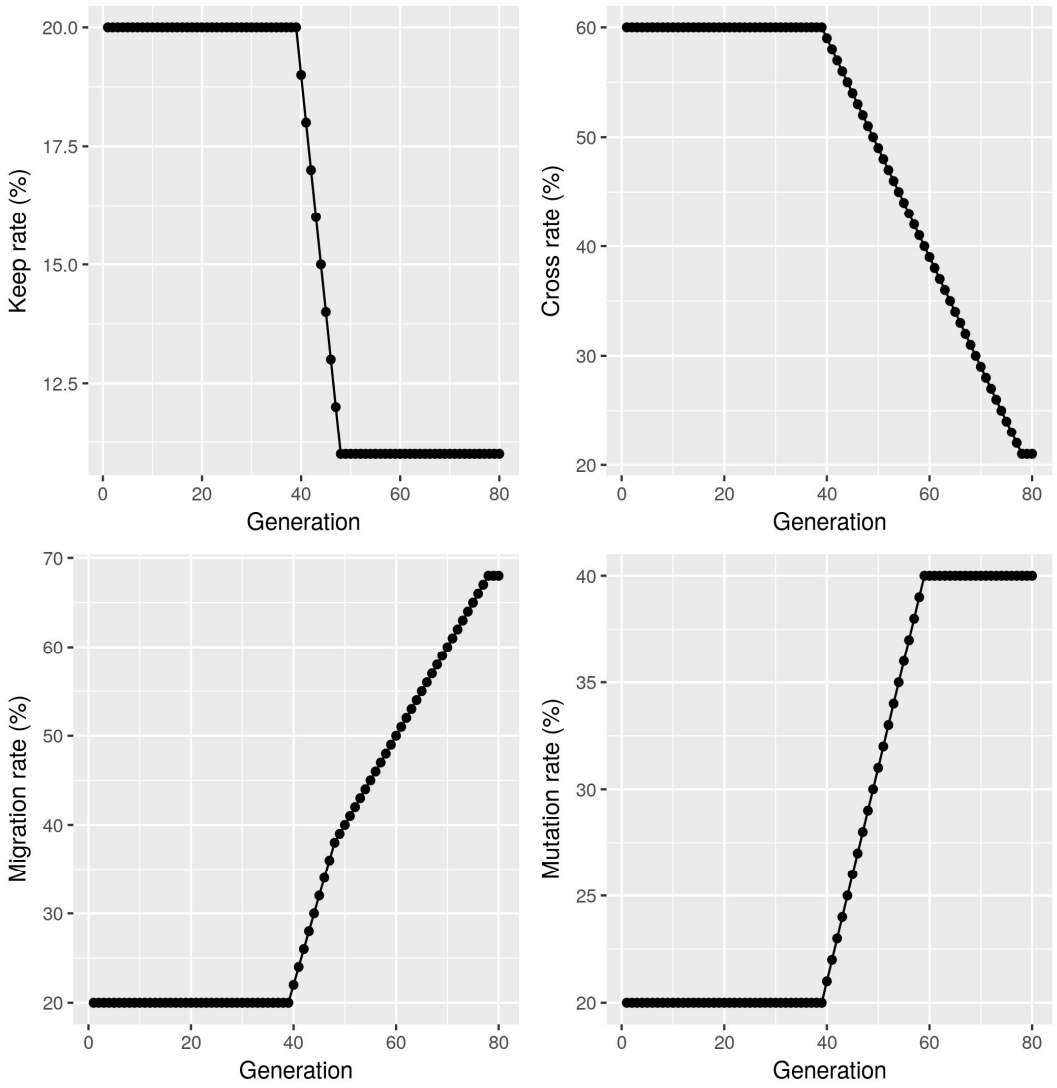

Figure 4.20 Changes in GA parameters: keep rate, cross rate, migration rate and mutation rate. The initial values for these parameters were respectively: 20%, 60%, 20%, and 20%. On generation 40, the GA started gradually changing the parameter values, to explore different parts of the solution space and find a better solution. Note that y axis in these plots is truncated.

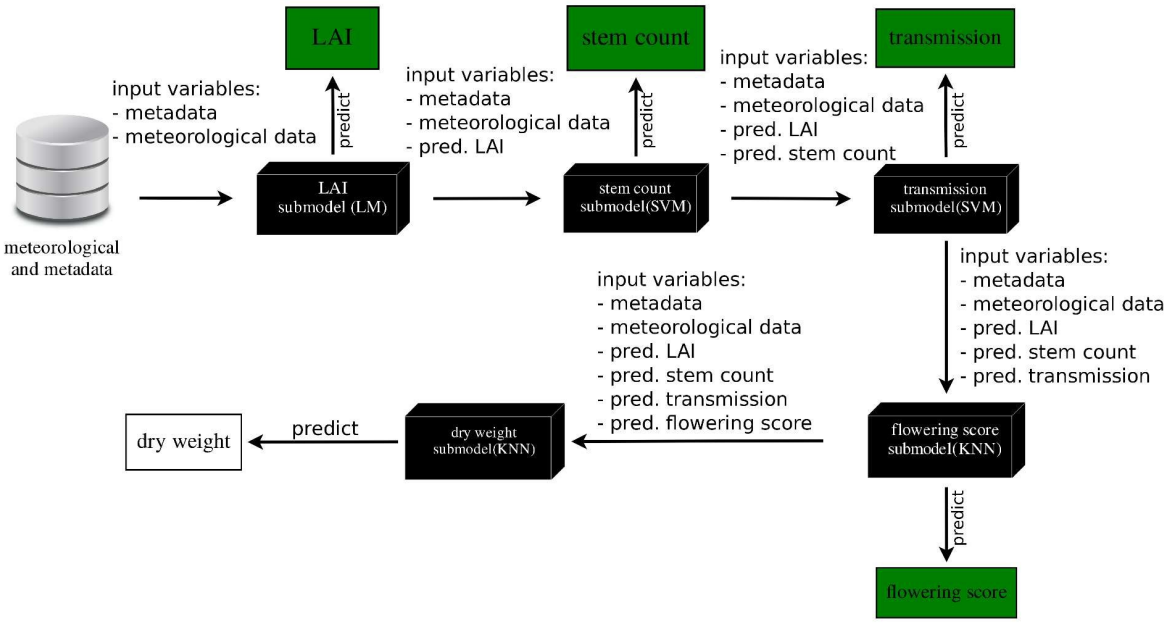

Figure 4.21 Structure of the best compound model found by the GA procedure. The canopy height variable was completely dropped from the model. Prediction starts with the LAI submodel (LM) which predicts LAI using just the model input data (meteorological and metadata). The flowchart is then followed through each submodel, adding all previously predicted variables to the input of the next submodel.

Testing model accuracy

The naïve model with the configuration shown on Figure 4.18 and the GA model as defined on Figure 4.21 were trained over the ABR61 training dataset (years 2011 – 2015). The two models along with APBPM were compared using RMSE and R^2 calculated using their dry weight predictions and real dry weight values from ABR61, year 2016. This test data set was not made available to the models during the submodel choice and structure optimisation process, and the training procedure that followed. The model results from the testing procedure are shown in Table 4.10, descriptive statistics for the dry weight from the test data (year 2016), are shown in Table 4.11.

Method	RMSE	R^2
APBPM	757	0.60
Naïve model	618.8	0.65
GA model	658.9	0.52

Table 4.10 Results from testing APBPM, naïve model and GA model against predicting dry weight in year 2016 from the ABR61 dataset.

Genotype	Mean	Standard deviation	Minimum value	25%	50%	75%	Maximum value
EMI-11	1631	713	551	928	1801	2066	3209
Giganteus	1620	929	485	641	1452	2190	3096
Goliath	1365	1043	258	468	1065	2135	3881
Sac-5	1684	763	641	978	1579	2244	3071

Table 4.11 Descriptive statistics for the dry weight values in the data against which the two compound models were tested – year 2016 from the ABR61 dataset.

The naïve model had the lowest RMSE compared to the other models, but it was closely followed by the GA model. However, the GA model explained the least amount of variance in the observed data ($\mathrm{R}^2 = 0.523$) compared to the rest of the models, and the naïve model explained the highest ($\mathrm{R}^2 = 0.651$). Besides the low R^2 value, the two models surpassed APBPM, producing more accurate yield predictions, as evidenced by their lower RMSE. As the three models were roughly on par, all of them were used in the work described by the next chapter.

4.4 Discussion

This chapter demonstrated the approaches used for building machine learning models of *Miscanthus* to predict biomass yield. Two categories of models: simple machine learning models and compound models, were examined separately. This ensured a fair comparison between models of similar functionality. Models from both categories were chosen for the next stage of the project, which determined the importance of collecting each variable and taking measurements at different times in the growing season for training accurate models.

Simple machine learning models

One of the categories, the simple machine learning models, contained single machine learning models that needed all phenotypic, meteorological and meta variables as input when producing yield predictions. Random forests, GBM and k-NN were demonstrated to be the best performers in this category, using cross-validation and measuring the prediction error RMSE. They also outperformed APBPM, although that comparison was not a fair one because APBPM accepted a different dataset, which had fewer variables but more data points. Using simple machine learning models for simulating crop growth from meteorological data was infeasible, as they required phenotypic data as well. Process models are usually used for this important task as they reduce the training phenotypic data into model parameters, and thus do not require input phenotypic data for making predictions.

However, because the simple machine learning models modelled the data well, the comparison served to demonstrate that the top three models were precise enough to reliably measure how and which perturbations in the training data would cause increase in prediction error, which was very important for the work described in the following chapter. There, perturbations of the training dataset were defined as the removal of a certain variable from training data, or even the removal of data points collected during a certain month of the growing season. The models were used to assess the impact of these perturbations on modelling accuracy, which identified important variables and months for training precise models.

An improvement of the models' accuracy was attempted, by applying bagging to each model, except for random forests. It offered limited accuracy improvement for GBM and k-NN which was outweighed by the additional time cost of training multiple models instead of one. These results led to the choice of random forests, and the standard versions of GBM and k-NN for use in the next chapter.

Compound models

The models in the second category reflected a more practical approach for *Miscanthus* modelling. They used only meta and meteorological variables to predict phenotypic variables and finally yield. In that respect, they resembled APBPM, and could be applied to simulating yield under different environmental conditions. A model preparation procedure was presented for each model. For the naïve model, this was the submodel choice – the choice of machine learning algorithms for each submodel. For the GA model, that was the submodel choice and the choice of model structure. The two models were shown to outperform APBPM, when tested against unseen data. This demonstrated that both models were fitting the data well and thus would also be useful for identifying important variables and months in the training data. Following these results, the naïve and GA models were chosen to be used in the next stage.

The compound models as well as APBPM had an important advantage over the manually parameterised model BPM (Davey *et al.*, 2015) described in the previous chapter. Once the parameterisation/training procedure for each model was programmed, which took some time, parameterising/training these models took very little time compared to BPM. This allowed the models, given the appropriate training data, to be trained for predicting yield for thousands of genotypes, which would be impractical for BPM.

Machine learning for crop modelling

An advantage common to all machine learning models described in this chapter is their ability to incorporate additional training data variables. Expanding APBPM with a new modelled phenotypic variable would mean that the model structure would have to change, as new equations would need to be added. With simple machine learning models, this is just a question of adding the variable to the training dataset

and instructing the training algorithm to use it. With compound models, some computational overhead will come from having to rerun the procedures for choosing submodels and model structure, but these methods could be programmed to be trained using any additional variables without having to make changes to the code each time.

The flexibility of machine learning models could prove useful as more data from various sources becomes available. Quite likely, recent advances in phenomics (Fahlgren, Gehan and Baxter, 2015), as well as the increased use of remote sensing data in crop modelling (Fang, 2003; Ines *et al.*, 2013), would result in higher dimensional training datasets for crop models in the future. The ability of machine learning to handle such data, and train accurate models quickly is an important advantage that could be instrumental for their increased use in crop modelling.

At this point, machine learning model interpretation is not straightforward, as models have been designed with prediction accuracy in mind. However, the application of GA in model structure optimisation has yielded an interesting result – canopy height was dropped from the model. While this result might be due to the limited size and diversity of the ABR61 dataset, and may not be replicable outside of this experiment, if GA is applied to a large and diverse dataset, its decisions on model structure may lead to interesting scientific questions on the interactions between variables and importance of different phenotypic variables to yield.

Additionally, compound models with their ability to predict plant growth (phenotypic variables) and yield just from meteorological data, could be applied to different climatology datasets, to simulate the performance of different genotypes in new environments. It would be a lot quicker to train them on datasets containing a large variety of genotypes and growing locations in different climates and meteorological conditions, compared to regular mechanistic models. Once trained on such data, they could become a very powerful tool for identifying genetic and environmental factors that influence plant growth and yield.

Chapter 5: Finding the optimal dataset for *Miscanthus* models using the scatter search approach

In this chapter a scatter search-based method was developed which looked for the optimal training dataset for building a predictive model for *Miscanthus* yield. The aim was to reduce the cost of data collection, while at the same time maintaining good modelling accuracy. The method evaluated the cost and benefit for each phenotypic and meteorological variable as well as the times of the season they were collected. The benefit was evaluated with respect to the accuracy of the models, trained with the machine learning methods selected in chapter 4. Importance scores were calculated for each variable and time of the season. This information could be used for planning the data collection phase of research projects, to allow for maximising the model accuracy and minimising the cost for obtaining the data.

5.1 Introduction

The identification of traits contributing significantly to crop yield is essential in collecting an informative dataset for the purposes of modelling and improving a crop. Yield is a complex trait in *Miscanthus* (Robson, Jensen, *et al.*, 2013) as well as in other crops (Frova *et al.*, 1999; Chapman *et al.*, 2003; Cuthbert *et al.*, 2008) and selecting for it directly is a challenging task. Yield is the result of the interaction between a number of secondary traits, including number of stems, plant height, stem thickness, and others. Identifying these and selecting for them, is an effective method for increasing the yield in a crop. Model simulations of crop yield benefit greatly from high quality phenotypic data, as it is vital for updating and refining the model (Craufurd *et al.*, 2013). Many mechanistic models simulate crop growth by taking into account underlying physiological processes like photosynthesis or radiation use efficiency, which in turn requires phenotypic data, particularly on leaf area index (LAI) (van Ittersum *et al.*, 2003; Poorter, Anten and Marcelis, 2013). However, collecting phenotypic data is time-consuming and costly, and acts as a bottleneck to breeding and modelling (Furbank and Tester, 2011). Thus, focusing on important phenotypic variables and finding other ways to remove unnecessary data is likely to increase the information content of the collected dataset and lower the costs for these processes.

A popular approach to test a phenotypic trait's contribution to yield is to look at the correlation coefficient between them. A study in switchgrass found plant height to have positive correlation with yield, whereas ash, nitrogen and acid detergent lignin content correlated negatively with yield (Lemus *et al.*, 2002). Another study in rice grain yield has found that measurements of flowering, plant maturity, height, stem count, and others correlated positively with yield (Sravan *et al.*, 2016). In *Miscanthus,* canopy height, stem count and stem diameter have been shown to be highly predictive of yield (Robson, Jensen, *et al.*, 2013; Kalinina *et al.*, 2017).

This approach is also used in studies looking for quantitative trait loci (QTL) – regions of the DNA that correlate with a phenotypic trait. Identifying yield-contributing traits and finding QTL for them provides valuable information for breeding programs, that could use the QTL to improve these traits and indirectly increase yield. For example, a QTL study in rice found the traits: panicles per plant, grain per panicle and 1000 grain weight to be the most correlated with yield (You *et al.*, 2006). In tropical maize, plant height, chlorophyll content of ear leaf, ears per plant, kernel number per plant and others have been identified as important for yield (Ribaut *et al.*, 2007). The QTL identified by the two studies as relevant for the phenotypic traits correlated with yield, are useful targets for future breeding using marker assisted selection.

Crop models also have a useful role in identifying yield contributing factors. Four models were trained in a study of important traits for sunflower grain oil concentration (Andrianasolo *et al.*, 2014): multiple linear regression, generalised additive model, regression tree and a simple sunflower oil concentration simulation model from a previous study (Pereyra-Irujo and Aguirrezábal, 2007). Three different methods were then used for finding variable importance: using the "calc.relimp" function from the "relaimpo" R package (Grömping, 2006), the variable importance measure of regression trees, and by calculating the Sobol indices (Sobol', 2001). Potential oil concentration was found to be the top contributor to sunflower grain oil, followed by leaf area duration and mean radiation use efficiency during post-flowering period. Another study in maize augmented its model training dataset with LAI and soil moisture remote sensing data, which had the effect of increasing the model accuracy (Ines *et al.*, 2013). Finally, a study in wheat used the Agricultural Production Systems Simulator (APSIM-N-wheat) model, which modelled daily plant growth and yield from weather data, soil

properties and nitrogen application. It described a low-cost way to map yield and simulate the effects of different rainfall, soil and nitrogen conditions. Using this method, it showed that the main contribution to grain yield was caused by an interaction between rainfall, soil plant available water capacity and fertiliser application. While this is not phenotypic data, the study demonstrated an approach for finding important variables for predicting yield (Wong and Asseng, 2006).

Another approach to increasing the information value of a dataset is to intelligently select the most informative samples. This has been applied in studies for agricultural classification using remote sensing data. For example, an SVM model for classifying winter wheat and spring barley was shown to perform better when trained against border training samples, then when trained using the whole dataset. Border training samples were data records which in the feature space were close to the border between classes (Foody and Mathur, 2004). The method was later applied to training SVM for more classes (cotton, basmati rice, local variety of rice, built up land and sand) with minimal loss of accuracy (Mathur and Foody, 2008). A more recent study applied intelligent data sample selection to normalised difference vegetation index (NDVI) remote sensing data (Zheng *et al.*, 2015). The selected samples were then used for training SVM on classifying nine major crop types, which resulted in an improved modelling accuracy, compared to models trained over the whole NDVI dataset.

Considering the information content of a dataset and the cost to collect the data presents an optimization problem – to find the optimal compromise between model accuracy and data collection cost. Usually, lower collection costs lead to a less informative dataset, which would diminish the crop model accuracy, unless an intelligent sampling strategy like those previously described is employed. There is a wide variety of methods for solving optimization problems. One possible family of methods for such tasks - evolutionary algorithms, is a set of population-based search algorithms that use evolutionary principles (selection, etc.) to navigate the search towards better solutions (Whitley *et al.*, 1996). One of the most well-known examples, genetic algorithms, have been a popular approach in crop models, as mentioned in section 4.1, mainly for finding optimal model parameters (Fang, 2003; Dai *et al.*, 2009; Klein *et al.*, 2012; Waongo *et al.*, 2013).

It has been suggested that GAs could be improved by using some of the strategies employed by another evolutionary algorithm: scatter search (Glover, 1994). Scatter search is a

population-based method, that works by maintaining a set of good solutions, called reference points, which are iteratively improved by applying a set of functions on them that explore beneficial changes to each solution. Besides improvement, the method aims to keep the solutions diverse, which allows it to explore more regions of the solution space (Fu, Glover and April, 2005). A modelling study of maize biomass and yield using remote sensing data, has used a scatter search-based method to optimise nine of its parameters (Battude *et al.*, 2016). The method used remotely-sensed green area index (GAI) and GAI predicted by the model, to calculate model root mean squared error (RMSE) and tuned the parameter values to minimise that RMSE. Validations of the trained model showed it performed well for estimating both biomass and yield.

This chapter describes a scatter search-based method for optimisation of *Miscanthus* data collection for the purposes of building predictive models. The aim was to solve the optimization problem of finding a way to reduce the cost of data collection, while at the same time maintaining good modelling accuracy. The method investigated the cost and accuracies associated with each variable from the dataset, as well as with the time of the growing season when measurements were taken, expressed as months. The method provided importance scores for each variable and month, which could be used for planning the data collection stage for new trials, where resources for collecting data are limited and maximum modelling accuracy is required.

5.2 Materials and methods

5.2.1 Overview

The method described in this chapter looked for the optimal data collection strategy which maximised modelling accuracy while minimising collection costs. Each strategy was defined by the months during which measurements would be collected, and the variables that would be measured. The data used in this chapter came from the ABR61 dataset, which was the same dataset used in the previous two chapters. Data from years 2011 to 2015 served as training data for models developed in the previous chapter. The method applied a scatter search optimisation algorithm that explored different data collection strategies and calculated the costs and model error associated with them. Collecting fewer data points or measuring fewer variables lowered the data cost. However, reducing the training data for

models meant that their accuracy was likely to be reduced. The scatter search algorithm looked for a strategy which achieved a good balance between cost and accuracy. It tested a number of strategies by using them to reduce the full training dataset and then train the models on the reduced data. The resulting model prediction accuracy and data collection cost (of the reduced dataset) were combined with a utility function – the model looked for strategies which minimised the utility value. An overview of the process is illustrated on Figure 5.1. The model errors of the best strategies found by the search algorithm were validated against year 2016 from the ABR61 dataset, which had not been made available to the models during the scatter search.

The cost of a data collection strategy reflected the cost of obtaining the data if that strategy was followed. Lower costs meant that fewer data points or variables were collected. The resulting data was used to train predictive models. Their average RMSE was the model error associated with the strategy. Scatter search was used to navigate through the strategy solution space, as evaluation of all possible strategies was infeasible. The algorithm looked for the optimal strategy which minimised the data collection cost and modelling error.

The models used to calculate the modelling error were developed in chapter 4. Two separate experiments were conducted using different sets of models. First, the scatter search method was run with three models from the simple machine learning group: k-nearest neighbour (KNN), random forests (RF), and generalised boosted models (GBM). The mean error of these three models was assigned as the model error associated with each strategy.

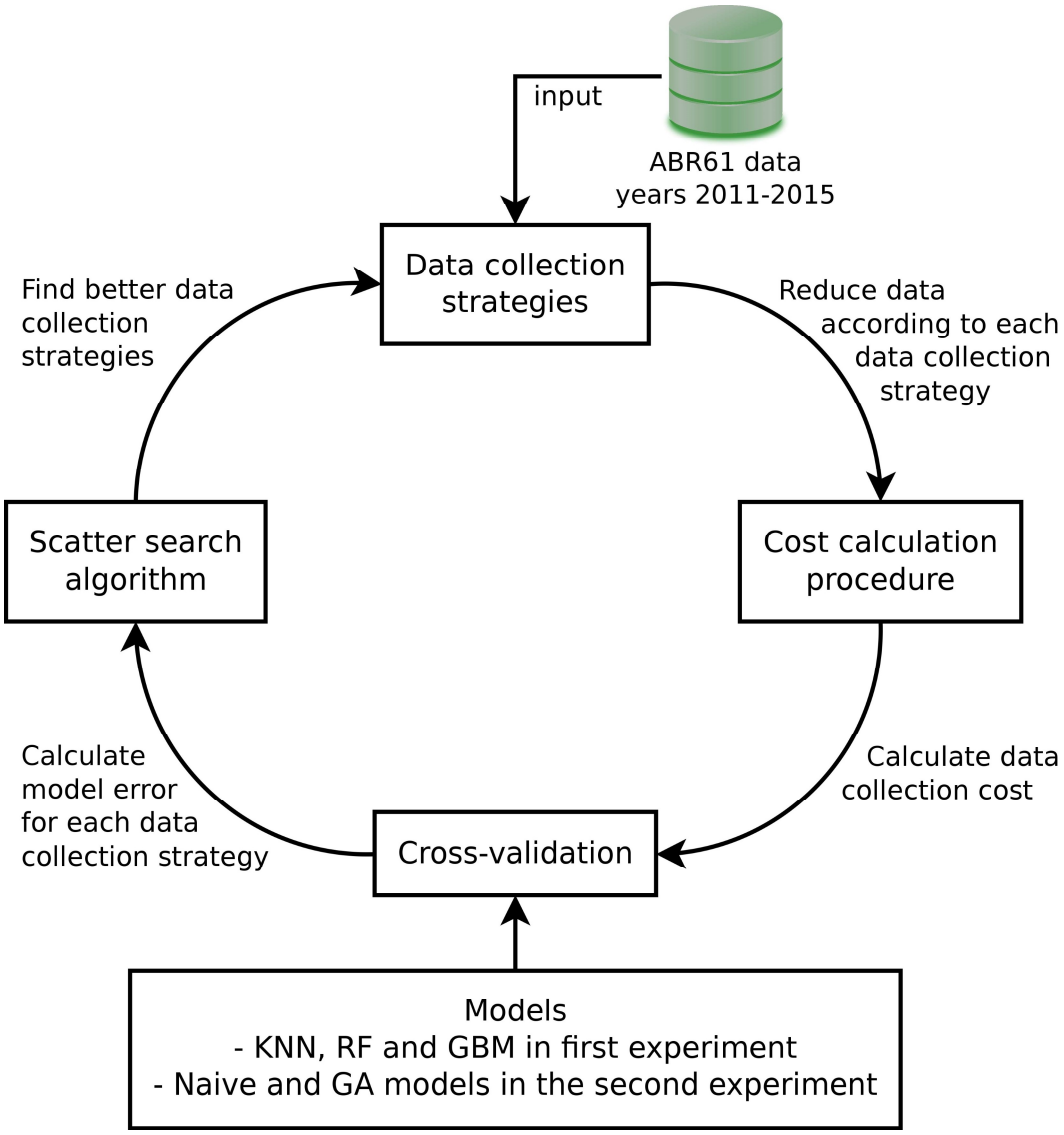

Figure 5.1 Overview of the scatter search-based method described in this chapter. The method started by randomly generating a set of data collection strategies. These contained instructions on which variables would be collected and during which months measurements would be taken (more information on data collection strategies can be found in section 5.2.2). Each data collection strategy was used to reduce the training data (dataset from ABR61 years 2011-2015). The cost for collecting each reduced dataset was calculated using the cost calculation procedure (also described in section 5.2.2). Cross-validation was used on each reduced dataset to calculate the model error associated with the corresponding data collection strategy. Using the strategy cost and error, the scatter search algorithm could explore new strategies and check if they bring an improvement over the existing ones (described in more detail in section 5.2.5).

In the second experiment, the same approach was followed, but two of the compound models developed in the last chapter were used instead: the naïve model and the GA model. The optimal submodel choice and model structure found in the previous chapter were kept for the two models. The calculation of model error is discussed in more detail in section 5.2.3. To test the accuracy of the results, the optimal strategy was validated against data from year 2016 from the same trial.

The following sections explain in more detail the methodology used in this chapter. First, in section 5.2.2 the meaning of data collection strategies is discussed, as well as the format they are presented in to the scatter search method, and the method used for calculating the cost for each strategy. Next, in section 5.2.3 the procedure for calculating model error is explained. The following section 5.2.4, discusses the utility function, a custom-defined function which combined the model error and strategy cost into a single number, indicating how good a data collection strategy was. The scatter search algorithm is described in section 5.2.5 and the validation procedure for the method is discussed in section 5.2.6.

5.2.2 Data collection strategies

Each data collection strategy contained instructions on the set of variables that should be measured and the periods of the growing season (defined by months), during which this should happen. The reason months were used is that they were the smallest time interval that was practical for presenting times in the season, with minimal gaps. A smaller interval, e.g. week, would have resulted in gaps (missing weeks), as no data had been collected during that specific time for the ABR61 dataset. Excluding a month from the data meant removing the rows containing measurements from that month from the data. For example, if a strategy stated to remove August, and the data was the sample data from Table 2.2, that would mean removing the rows with date falling within August, i.e. rows 3 and 8. The cumulative meteorological values of the other rows did not change, so they still contained the amounts which were accumulated during the now missing month of August.

Table 5.1 illustrates an example data collection strategy, where the variables: canopy height, transmission, row, genotype, day of year, and PAR, would be measured during the months: June, July, and October. Whether a month or variable was used by a strategy depended on the Boolean value associated with it (also shown on Table 5.1). The list of variables and

months shown on Table 5.1 are referred to collectively as the strategy parameters. To calculate the cost of a strategy, cost values were assigned to each variable, which reflected the time (in seconds) required to collect each variable by measuring a single plant. All strategy parameters, as well as their associated costs, where applicable, are shown on Table 5.2.

The metadata and meteorological variables had 0 cost because they were automatically recorded. As each plant had a unique identification (UID), the values for genotype, row and column were immediately known, because they were associated with that UID. Measurement date was recorded automatically by the software when phenotypic measurements were recorded. Finally, meteorological variables were collected automatically by the nearby meteorological station.

Name	Use
May	False
June	True
July	True
August	False
September	False
October	True
Stem count	False
Canopy height	True
Transmission	True
Flowering score	False
Leaf area index (LAI)	False
Row	True
Column	False
Genotype	True
Day of year	True
Degree days	False
Rainfall	False
Photosynthetically active radiation (PAR)	True

Table 5.1 An example data collection strategy. For each month (May – October) the Boolean variable in the "Use" column defined whether measurements would be taken during that time. For variables, the "Use" column defined whether the corresponding variable would be measured. Phenotypic variables were not summarised, instead the actual measurements on each measurement date were used for the analysis in this chapter. Meteorological data were comprised of the accumulated values since plant emergence, as shown in Table 2.2 in chapter 2.

Name	Type	Cost
May	Month	NA
June	Month	NA
July	Month	NA
August	Month	NA
September	Month	NA
October	Month	NA
Stem count	Phenotypic variable	27.1
Canopy height	Phenotypic variable	21.4
Transmission	Phenotypic variable	28
Flowering score	Phenotypic variable	10.4
Leaf area index (LAI)	Phenotypic variable	124.5
Row	Metadata variable	0
Column	Metadata variable	0
Genotype	Metadata variable	0
Day of year	Metadata variable	0
Degree days	Meteorological variable	0
Rainfall	Meteorological variable	0
Photosynthetically active radiation (PAR)	Meteorological variable	0

Table 5.2 Costs associated with each strategy parameter, except for months, as no costs were set for individual months. The data presented in this table were used for calculating the costs of each data collection strategy. The cost was calculated by multiplying the sum of all measured variables by the number of months during which data was collected. This is the reason why no cost was associated with month entries. Meteorological and metadata variables had 0 costs because the data was collected automatically. The assigned costs, particularly for the meteorological variables, were naïve – they did not account for the upfront cost of the station, or for the labour and expenses for its operation, maintenance and data offloading. In the future, this method could be improved significantly by assigning the real financial costs of recording each variable.

The method used for calculating strategy cost did not require month costs, so costs were only assigned to variables. To calculate each strategy cost, the sum of the cost values of each collected variable was multiplied by the number of months collected as shown in the equation below:

$$\mathrm{cost}_{\mathrm{absolute}} = \mathrm{months} \times \sum_{\mathrm{i}=0}^{\mathrm{n}} (\mathrm{cost}_{\mathrm{i}} \times \mathrm{use}_{\mathrm{i}})$$

Where:

- months was the number of months
- $\mathrm{use}_{\mathrm{i}}$ was 0 if variable i was not collected and 1 otherwise
- $\mathrm{cost}_{\mathrm{i}}$ was the cost value for variable i taken from Table 5.2

The number of measurements varied between months and are shown on Figure 5.2. As measurements damaged the plots, especially in late season, when the plots were dense and the plants have reached their maximum growth, measurements in September and October had to be reduced. The different number of measurements in each month meant that there was a caveat in the cost function, which only considered the number of months being measured and not which ones. As there was no straightforward way to compensate for that, this caveat was considered when discussing the results. However, this caveat had a minor influence, as becomes clear from the explanation in section 5.2.4, with further evidence in the results of the two experiments: sections 5.3.1 and 5.3.2.

Each strategy's cost value was converted from the absolute cost, as defined in the equation above, to a value relative to a reference cost. The reference cost $\mathrm{cost}_{\mathrm{ref}}$ was the absolute cost of the reference data collection strategy which had all strategy parameters set to True, i.e. no variables or months were removed from the full dataset.

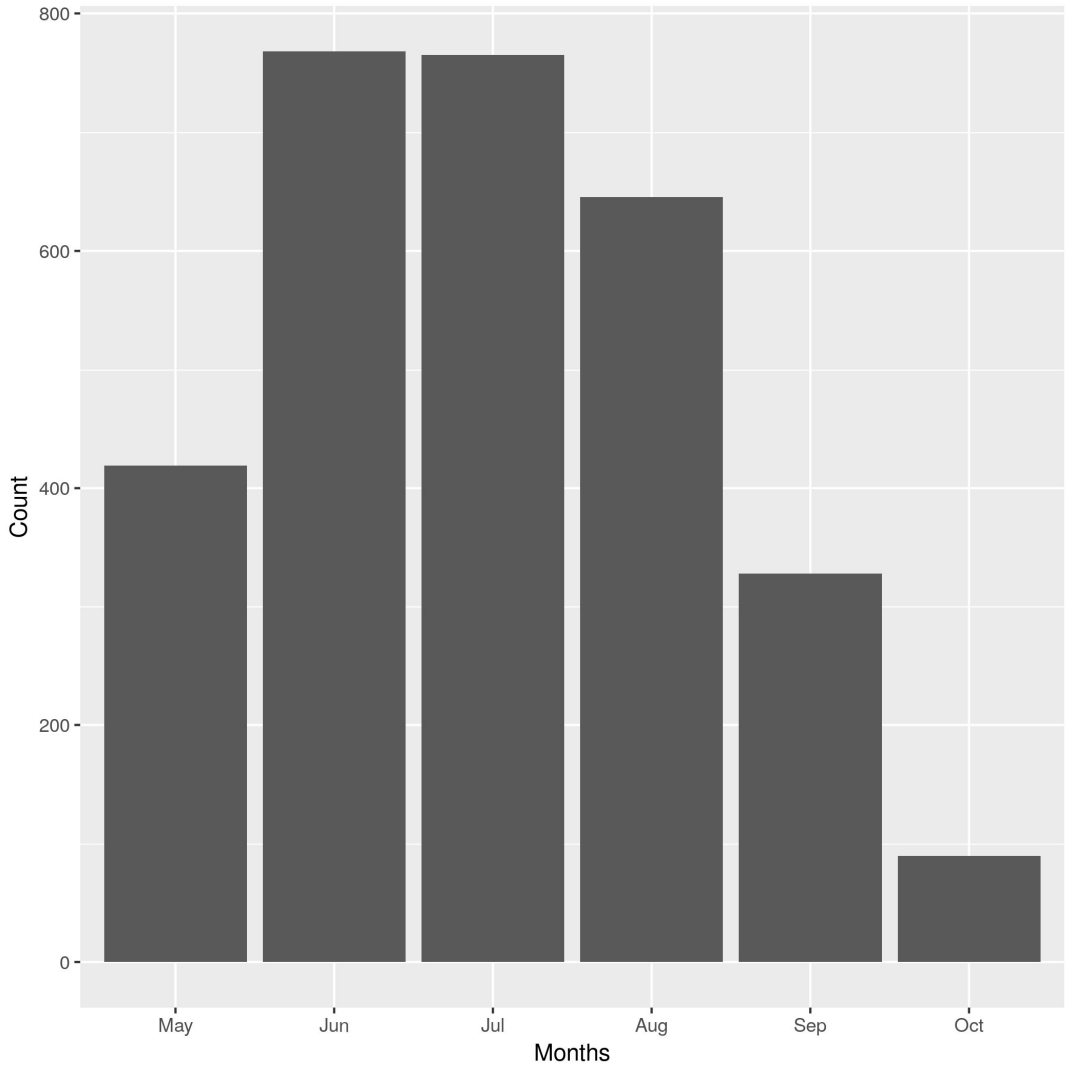

Figure 5.2 Number of measurements in each month in the ABR61 dataset, years 2011-2015. This was the dataset used by the scatter search to estimate model error for each data collection strategy. The number of measurements varied between months, but the data collection cost formula did not reflect that. However, this problem was a minor caveat, as discussed in section 5.2.4.

Because of that it was also the maximum possible cost of any collection strategy. The relative cost $\text{cost}_{\text{strategy}}$ was calculated using the formula below:

$$\text{cost}_{\text{strategy}} = \frac{\text{cost}_{\text{absolute}}}{\text{cost}_{\text{ref}}}$$

This scaled the absolute cost to a number between 0 and 1. As the variable name suggests, the relative cost was assigned as the cost of the strategy. Strategies with $\text{cost}_{\text{strategy}} = 1$ had the maximum possible cost, whereas those with $\text{cost}_{\text{strategy}} = 0$ had the minimum. Strategies that omitted the phenotypic variables had relative cost of 0, as meteorological and metadata variables had 0 cost. Relative cost was preferred to absolute cost, because it facilitated its combined use with another important metric of data collection strategies – the model error. Section 5.2.4 contains a more detailed explanation of how these two metrics were used together.

5.2.3 Model error

While the strategy cost served as a measure for one side of the data collection optimisation problem – the need to reduce the cost for obtaining data, the model error metric presented the other side – the objective to still develop accurate models. The error associated with a data collection strategy was calculated using cross-validation over a set of models. The data used for this purpose was the ABR61 dataset, years 2011 – 2015. The approach closely resembled the one used for screening simple machine learning models in chapter 4. The main difference in the cross-validation used in this stage was that the data collection strategy was applied to the training data in each fold, which reduced the amount of data available to the models for training. This simulated the reduced dataset that following the data collection strategy would have produced. For the example strategy shown in Table 5.1, the training data for each cross-validation fold would have been reduced, by removing the variables: stem count, flowering score, LAI, column, degree days, rainfall, as well as removing all data points collected in the months: May, August, September.

Simple machine learning model experiment

The data used for the two experiments also differed, because of the difference in the input data accepted by models from the two categories. The models used in this experiment were

the best three from the simple machine learning models category. These were KNN, RF and GBM, as established by the model comparison in the previous chapter.

The models in the simple machine learning models category could only work with data records that contained dry weight measurements, as this was their prediction target. This meant that only years 2011 and 2015 were used in this case. Table 5.3 lists the training and test datasets for each fold of the first experiment. Cross-validation fold 2 is illustrated in Figure 5.3 and the following text describes it in more detail, but the procedure for fold 1 is equivalent. In fold 2 the training dataset consisted of data from year 2015. It was reduced by the strategy and the three models (KNN, RF and GBM) were trained on it. The test dataset, which in fold 2 consisted of year 2011 from ABR61, was provided as input to each model, which then made predictions over it. The test data was not reduced by the strategy, as the full test data was necessary. This ensured comparability between the RMSE values of different strategies, because the models were always tested against the same full data set. If instead the test data was also reduced, it would have meant that some models would have been tested against fewer data points than others. Potentially, this could have made some strategies look unrealistically better than others. The same process was used for fold 1, but year 2011 was used for training and year 2015 for testing.

RMSE was used for measuring model prediction error. It was calculated individually for each model, in the same way as described in section 4.2.3. The model's dry weight predictions, as well as the real dry weight values were pooled together and the RMSE formula was applied. This gave the absolute RMSE value ($\mathrm{RMSE}_{\mathrm{absolute}}$) of each model. However, to use the RMSE

Fold	Model	Training data	Test data
1	KNN	ABR61 reduced, year 2011	ABR61 full, year 2015
1	Random forests	ABR61 reduced, year 2011	ABR61 full, year 2015
1	GBM	ABR61 reduced, year 2011	ABR61 full, year 2015
2	KNN	ABR61 reduced, year 2015	ABR61 full, year 2011
2	Random forests	ABR61 reduced, year 2015	ABR61 full, year 2011
2	GBM	ABR61 reduced, year 2015	ABR61 full, year 2011

Table 5.3 Training and test data used for each fold and model in the cross-validation of the first scatter search experiment. As it involved using simple machine learning models, only data from years 2011 and 2015 could be used, as these models could not use records without values for dry weight. The training data was reduced using the data collection strategy that was being evaluated. Test data was not reduced, as this ensured comparability between strategies' RMSEs.

alongside the relative cost, a relative RMSE value was needed. The cost value of each strategy was set to be relative to the maximum possible cost – that of the default strategy, which had all strategy parameters set to True. This cost calculation method was described in section 5.2.2. Following similar logic, the RMSE of each model was set to be relative to the reference RMSE ($\mathrm{RMSE}_{\mathrm{ref}}$), obtained by cross-validation of a reference model, trained by the same machine learning algorithm (KNN, RF or GBM) over the unreduced dataset (ABR61 years 2011 and 2015). For example, to calculate the relative RMSE of a KNN model which was trained on a reduced dataset, the $\mathrm{RMSE}_{\mathrm{absolute}}$ was calculated and used alongside the $\mathrm{RMSE}_{\mathrm{ref}}$ of a reference model, also trained by KNN over the unreduced dataset.

Minimum and maximum RMSE values ($\mathrm{RMSE}_{\mathrm{min}}$ and $\mathrm{RMSE}_{\mathrm{max}}$) were needed for scaling $\mathrm{RMSE}_{\mathrm{absolute}}$ between 0 and 1. It was expected that the reference RMSE would have been the lowest possible RMSE value, because the model was given the highest amount of information. Thus, initially the scatter search algorithm was run with the min and max values set to:$\mathrm{RMSE}_{\mathrm{min}} = \mathrm{RMSE}_{\mathrm{ref}}$ and $\mathrm{RMSE}_{\mathrm{max}} = 2 \times \mathrm{RMSE}_{\mathrm{ref}}$. The maximum RMSE value was arbitrarily set to be twice the reference RMSE. The lower the value of $\mathrm{RMSE}_{\mathrm{max}}$ the higher the penalty on strategies with RMSEs higher than that, as the scatter search algorithm sought to lower both cost and error values. The value of $2 \times \mathrm{RMSE}_{\mathrm{ref}}$ was an initial guess of what the optimal $\mathrm{MSE}_{\mathrm{max}}$ should be. It caused the scatter search method to be unwilling to consider strategies that doubled the prediction error. The relative RMSE for the model was calculated using the Min-Max normalisation formula (Mohamad and Usman, 2013) below:

$$\mathrm{RMSE}_{\mathrm{relative}} = \frac{\mathrm{RMSE}_{\mathrm{absolute}} - \mathrm{RMSE}_{\mathrm{min}}}{\mathrm{RMSE}_{\mathrm{max}} - \mathrm{RMSE}_{\mathrm{min}}}$$

After the relative error values were calculated for each model, their average was assigned as the data collection strategy error:

$$\mathrm{RMSE}_{\mathrm{strategy}} = \frac{\mathrm{RMSE}_{\mathrm{knn\ relative}} + \mathrm{RMSE}_{\mathrm{rf\ relative}} + \mathrm{RMSE}_{\mathrm{gbm\ relative}}}{3}$$

Using the average value and not each individual model's RMSE, made it considerably easier to use model error alongside strategy cost when deciding which strategies were better, as illustrated in section 5.2.4.

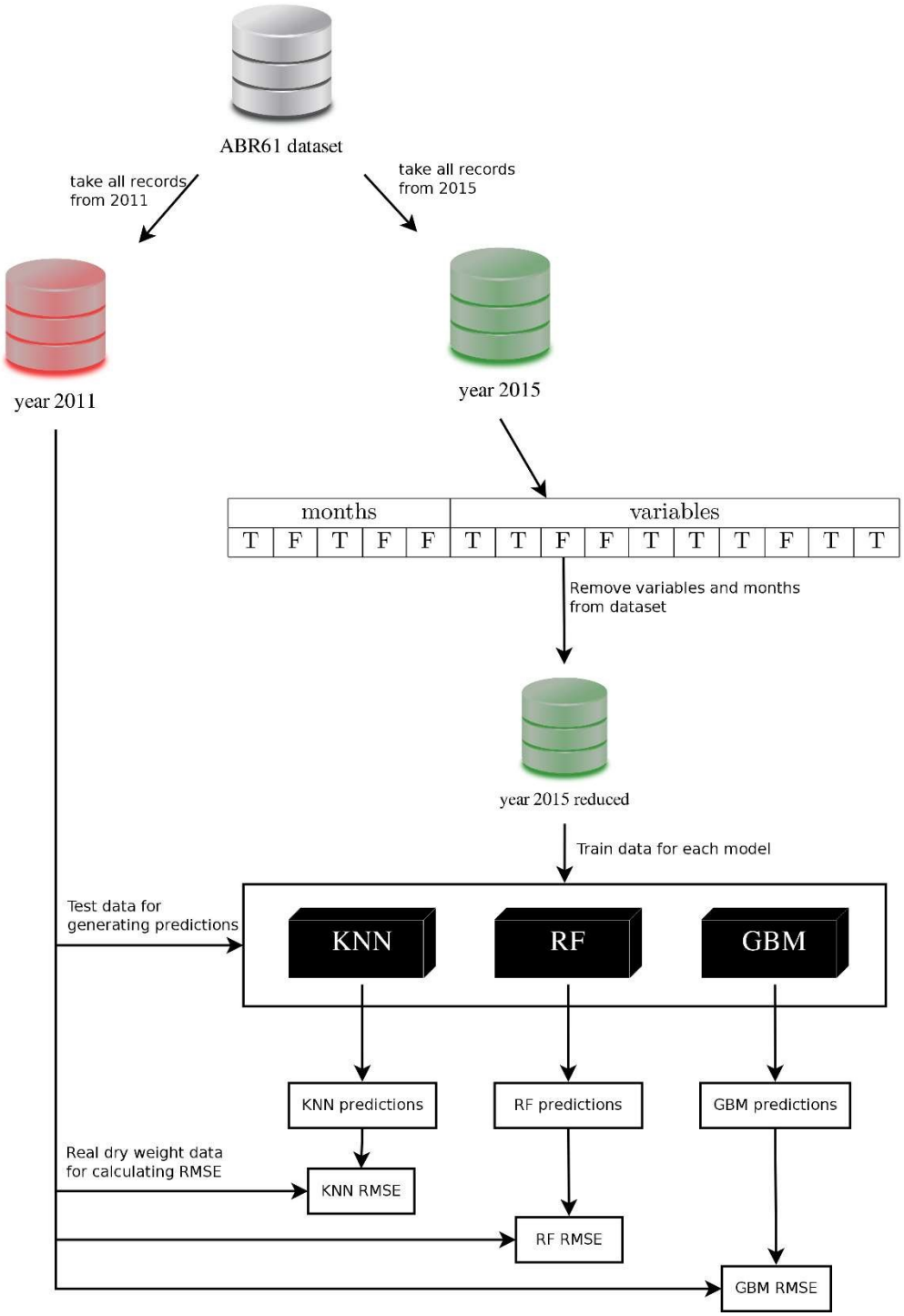

Figure 5.3 Fold 2 of the first scatter search experiment. The training data (year 2015 from the ABR61 dataset) was reduced by the data collection strategy, to include only data that would have been collected following the strategy. Each model was trained on the reduced dataset. Test data from year 2011 was then provided as input to each model to generate predictions. The real dry weight values from the test data and the predictions were used to calculate RMSE for each model.

An initial test run of the scatter search algorithm generated above 20,000 unique strategies. For each strategy, the RMSE of each of the three models was calculated. An inspection of the histogram of the $\mathrm{RMSE}_{\mathrm{absolute}}$ values for each model revealed that the maximum and minimum values were inadequate. The scatter search algorithm produced data collection strategies whose models had lower RMSE than the assumed minimum. This caused negative $\mathrm{RMSE}_{\mathrm{ref}}$ values. Additionally, it was observed that the maximum RMSE was set too high, and reducing it would have improved the scatter search algorithm's sensitivity to strategies within a closer range around $\mathrm{RMSE}_{\mathrm{ref}}$. The maximum RMSE was set to $\mathrm{RMSE}_{\mathrm{max}} = 1.2 \times \mathrm{RMSE}_{\mathrm{ref}}$, which produced an upper boundary placed below high error outliers, making it more unlikely for the scatter search method to prefer one of them. The number 1.2 was found to be optimal by using the RMSE histogram and a trial-and-error approach. The lower boundary ($\mathrm{RMSE}_{\mathrm{min}}$) was set individually for each model, as each model's RMSE distribution was different. A formula, that used the smallest error found by the scatter search for each model ($\mathrm{RMSE}_{\mathrm{best}}$), was developed to calculate the minimum RMSE:

$$\mathrm{RMSE}_{\mathrm{min}} = \frac{\mathrm{RMSE}_{\mathrm{ref}}}{(\frac{\mathrm{RMSE}_{\mathrm{ref}}}{\mathrm{RMSE}_{\mathrm{best}}} + 0.1)}$$

$\frac{\mathrm{RMSE}_{\mathrm{ref}}}{\mathrm{RMSE}_{\mathrm{best}}}$ gave the ratio between the reference RMSE and the best (smallest) RMSE found by the initial run of the scatter search. Dividing $\mathrm{RMSE}_{\mathrm{ref}}$ by that ratio would have simply given $\mathrm{RMSE}_{\mathrm{best}}$. Since it was expected that scatter search would find better strategies with $\mathrm{RMSE}_{\mathrm{absolute}} < \mathrm{RMSE}_{\mathrm{best}}$, when run for a longer period, the lower boundary was shifted to the left ($\mathrm{RMSE}_{\mathrm{min}}$ was made slightly smaller) to accommodate such values. This was achieved by the 0.1 term in the equation above. If this had not been done, the $\mathrm{RMSE}_{\mathrm{relative}}$ values of strategies with smaller error than the current best, would have been negative.

While this approach could not actually guarantee that the algorithm would not find a strategy which generated error lower than the minimum, it established a lower boundary that was below all observed error values up to that point.

Compound model experiment

In the second experiment, the two compound models (naïve and GA model) developed in the previous chapter were used in place of the simple machine learning models from the previous experiment. The optimal submodel and model structure choices were taken from the results

of the previous chapter and cross-validation was done on the two models. As the models were compound models it was possible to use data from years 2012 and 2014 alongside the data from 2011 and 2015, because of the ability of the compound models to make use of records without dry weight measurements. More specifically, the records that lacked dry weight measurements, were still useful for training the phenotypic submodels. As was the case for simple machine learning models, the dry weight submodels could not use records without dry weight data for training.

The cross-validation folds still had to be two, because dry weight predictions had to be compared to real measurements, which were only available for 2011 and 2015. Thus, the test data only consisted of data from these two years as shown on Table 5.4, which lists the training and test data for each fold. Data from 2012 and 2014 was thus added only to the training data of each fold. Beside the changes in data and models, the same procedure illustrated on Figure 5.3 was followed for cross-validating compound models.

Fold	Model	Training data	Test data
1	Naïve model	ABR61 reduced, years 2011, 2012, 2014	ABR61 full, year 2015
1	GA model	ABR61 reduced, years 2011, 2012, 2014	ABR61 full, year 2015
2	Naïve model	ABR61 reduced, years 2012, 2014, 2015	ABR61 full, year 2011
2	GA model	ABR61 reduced, years 2012, 2014, 2015	ABR61 full, year 2011

Table 5.4 Training and test data used for the two compound models in each fold of the cross-validation of the second scatter search experiment. The experiment used the naïve model and the GA model, so it was possible to include the data from years 2012 and 2014 for training each model.

Finding the right values for $\mathrm{RMSE}_{\min}$ and $\mathrm{RMSE}_{\max}$ presented the same challenges as in the previous experiment. The initial values of $\mathrm{RMSE}_{\min} = \mathrm{RMSE}_{\mathrm{ref}}$ and $\mathrm{RMSE}_{\max} = 2 \times \mathrm{RMSE}_{\mathrm{ref}}$ were inadequate. A histogram of RMSE from nearly 7000 strategies was used to verify that setting the minimum and maximum to:

$$\mathrm{RMSE}_{\min} = \frac{\mathrm{RMSE}_{\mathrm{ref}}}{(\frac{\mathrm{RMSE}_{\mathrm{ref}}}{\mathrm{RMSE}_{\mathrm{best}}} + 0.1)}$$

and:

$$\mathrm{RMSE}_{\max} = 1.4 \times \mathrm{RMSE}_{\mathrm{ref}}$$

produced useful boundaries. The 1.4 value was found by using the histogram and a trial-and-error approach. A lower number of strategies was used for this experiment, as cross-validation of compound models took significantly longer compared to the simple machine learning models. This was because it took considerably longer to train compound models compared to the simple machine learning models. To illustrate this point, each model (RF, KNN, GBM, naïve and GA model) was trained against the ABR61 dataset, years 2011-2015. This was repeated 50 times for each model. The time it took for each model to complete all 50 trainings is shown in Table 5.5. Thus, a smaller number of strategies could be generated when cross-validation was done with the naïve and GA models.

Name	Random forests	K-nearest neighbour	Generalised boosted models	Naïve model	GA model
50 models train time (HH:MM:SS.F)	00:00:11.5	00:00:01.6	00:00:04.5	00:18:06.2	00:33:04.2

Table 5.5 Time required for each type of model to be trained 50 times. Each model was trained 50 times on the ABR61 dataset, years 2011-2015. The simple machine learning algorithms (RF, KNN and GBM) completed a single training for a fraction of a second which is why training was repeated 50 times. The times are expressed in the HH:MM:SS.F format (hours, minutes, seconds and fractions of a second).

$\mathrm{RMSE}_{\mathrm{relative}}$ was calculated using the same formula from the previous experiment. $\mathrm{RMSE}_{\mathrm{strategy}}$ was calculated in an analogous way:

$$\mathrm{RMSE}_{\mathrm{strategy}} = \frac{\mathrm{RMSE}_{\mathrm{naive\ relative}} + \mathrm{RMSE}_{\mathrm{GA\ relative}}}{2}$$

While this experiment made use of only two models, instead of three as in the previous one, it was more important to calculate $\mathrm{RMSE}_{\mathrm{strategy}}$ from well performing models. The naïve and GA models had lower error when tested against APBPM in the previous chapter. The application of less accurate models would have led to a bad estimation of model error, which would have influenced the results from the scatter search method badly. Thus, it was preferred to test strategy RMSE with a few accurate models, rather than many inaccurate models.

5.2.4 Utility function and algorithm choice

The two statistics described in the previous sections $\mathrm{cost}_{\mathrm{strategy}}$ (section 5.2.2) and $\mathrm{RMSE}_{\mathrm{strategy}}$ (section 5.2.3), needed to be combined into a single value, called utility, which

represented how good a strategy was. The optimisation algorithm compared data collection strategies by their utility values, to find the one that minimised the cost and error. The utility was calculated using the function below:

$$\text{utility} = 0.1 \times \text{cost}_{\text{strategy}} + \text{RMSE}_{\text{strategy}}$$

The cost function was multiplied by the scaling parameter 0.1, to put higher priority on minimising the modelling error over minimising the cost. This put a higher priority on finding data collection strategies that yielded accurate models, compared to cheap strategies. The reason for this scaling was that small changes in the cost were not considered as important as small changes in the model error.

The 0.1 scaling parameter had a diminishing effect on the caveat that the cost function did not consider number of measurements for each month, as described in section 5.2.2. That was caused by the smaller role that cost had to play for influencing the utility value and subsequently making one strategy be considered better than another. The 0.1 scaling factor was picked by examining the usual values for $cost_{strategy}$ and $RMSE_{strategy}$ and then picking a number that balanced them well. A better approach would be to normalise the cost and RMSE values and assign scaling parameters to both. In that case it would be easier to gain an intuitive understanding of the balance between cost and error that the utility function aims to find.

The definitions of data collection strategy, cost, modelling error and the utility function can be used for formally defining the problem addressed in this chapter: given the set of strategy parameters listed under Table 5.1, that could take binary values (True or False), find the optimal solution, that minimise the value of the utility function. Finding the solution (strategy) involved picking the right values for each parameter. Their binary (True/False) nature made this problem a binary optimisation problem.

As the utility function had no defined derivative, it was impossible to use any derivative-based hill climbing approaches (Tsoumakas, Partalas and Vlahavas, 2009). Another popular approach for similar problems, binary integer programming (Theunissen, 1985), was also infeasible, due to the substantial number of viable strategies that need to be evaluated using the utility function. The remaining option was a black box approach, where strategies were generated, evaluated and improved in a stepwise manner. One such method, made

specifically for binary problems is the black box scatter search algorithm (Gortázar *et al.*, 2010).

5.2.5 Black box scatter search for binary optimization problems

Algorithm overview

Black box scatter search is an evolutionary algorithm, which evolves a set (called reference set) of good solutions called reference points, and applies a set of methods to improve these solutions. A solution in the context of the method described in this chapter meant a data collection strategy. In this section, the two terms are used synonymously.

The foundation of the algorithm consisted of four categories of methods:

- Diversification generation:

This category contained three different methods of generating diverse solutions. Diversity between solutions (data collection strategies) was determined by their Hamming distance (Steane, 1996), which was the number of parameters for which they did not have the same value (True/False). For example, the most different strategy from the example one shown in Table 5.1 would be a strategy which takes the complete inverse of the parameter values (True where False and vice versa). The three methods in this category generated an n number of different strategies, which served as a starting point for the search process.

- Improvement:

A method for improving each solution by changing the value of each variable in turn and recalculating the utility function. It allowed the algorithm to look at each solution's immediate neighbours for ones with improved utility. For example, for the solution in Table 5.1, it would create a new strategy where value for "May" is True and calculate its utility. Then it would create another from the original where "June" is False (so "May" is still False) and calculate its utility. It would repeat that process for all possibilities.

- Combination

Combination contained seven different methods of combining solutions together to create a better solution. For example, the first method took the union of two strategies. For the purposes of this example, two strategies $x = \{x_1, x_2\}$ and $y = \{y_1, y_2\}$ are assumed, with

only two parameters, where each parameter x_i and y_i could be True or False. The union of these strategies z would be:

$$z_i = 1 \; if \; x_1 = 1 \; and \; y_i = 1$$
$$z_i = 0 \; otherwise$$

- Reference set update:

The selection method at each iteration of the algorithm – it balanced between selecting the best solutions and the most diverse solutions. Half of the reference set was filled with the best strategies. The other half was filled with solutions that were diverse from the ones already there.

Describing the exact implementation of these methods is outside of the scope of this chapter, which only provides an overview of the algorithm. More information on the specifics of each method can be found in the original publication (Gortázar *et al.*, 2010). To run, the algorithm required two algorithm parameters to be specified. They had some influence over the algorithm's functionality:

- b – this parameter specified the size of the reference set of solutions. For the simple machine learning model experiment $\mathrm{b} = 20$, and for the compound model experiment, this was set to $\mathrm{b} = 8$, due to the significant overhead of training the compound models.
- $\mathrm{population\_size}$ – specified the size of the initial population of solutions from which the reference set was formed. For the simple machine learning experiment, $\mathrm{population\_size} = 45$, but for the compound model experiment, it was lowered to $\mathrm{population\_size} = 12$, for the same reason as the b parameter.

The algorithm placed solutions in three different containers: the reference set, the population and the database, each with a different purpose. Half of the reference set contained the best solutions and the other half was filled with solutions as different as possible from the ones already there. The population was a set of diverse solutions from which the reference set was built in the first iteration. The population can be thought of as a starting point of the algorithm. Finally, the database contained all solutions encountered by the algorithm. The algorithm checked the database each time it generated a new solution, which prevented it from calculating the error and cost for the same solution twice.

The algorithm is summarised below. For more details please refer to (Gortázar *et al.*, 2010).

1. A starting population of diverse solutions (data collection strategies) was generated using the diversification generation methods. The size of the population was $\mathrm{population\_size}$. The utility value for each solution was calculated, by calculating its cost and error. For calculating the cost, the method for calculating data collection strategy cost described in section 5.2.2 was used. For calculating the error, the cross-validation method in section 5.2.3 was applied, for finding the RMSE of either the simple machine learning models or the compound models, depending on the experiment.
2. The best $\frac{\mathrm{b}}{2}$ solutions were improved using the improvement method and added to the reference set. The factor determining the goodness of each solution was the utility value. The best solutions were the ones with the lowest utility.
3. Another $\frac{\mathrm{b}}{2}$ solutions were chosen from the population, such that they were most distant (or diverse, measured by Hamming distance) from the solutions already in the reference set. These solutions were added to the reference set.
4. A pool of new solutions (denoted by pool) was created, using each possible combination of solutions from the reference set. A new solution was created from a combination using a randomly picked method from the seven combination methods. Initially the methods had the same probability to be picked, but overtime, methods that produced better solutions had their probability increased. The utility value was calculated for each new solution following the methodology described in sections 5.2.2, 5.2.3, and 5.2.4).
5. A new reference set was created with the best $\frac{\mathrm{b}}{2}$ solutions from the joint between the two sets: $\mathrm{reference\ set} \cup \mathrm{pool}$.
6. The most diverse $\frac{\mathrm{b}}{2}$ solutions from the joint set were added to the new reference set.
7. Steps 4-6 were repeated until the reference set stopped changing or the maximum number of iterations had been reached.

Score table

Each time the algorithm generated a new solution (data collection strategy), it added it to a database of seen solutions. This allowed the method to build a score table for strategy

parameter scores, which reflected the likely impact of setting a strategy parameter (e.g. stem count, LAI, June) to True or False on the utility value. The score for a parameter x_i taking the value True was calculated using the formula:

$$\text{score}_T(i) = \frac{\overline{f_T}}{(\overline{f_T} + \overline{f_F})}$$

Where:

- $\overline{f_T}$ was the mean utility value for all encountered solutions where $x_i = \text{True}$
- $\overline{f_F}$ was the mean utility value for all encountered solutions where $x_i = \text{False}$

Both score_T and score_F were calculated when building the score table. Besides the optimal solution to the optimization problem, the score table was another useful output from the scatter search method. The value of the parameter score showed the average impact of a parameter on the utility value. For example, if $\text{score}_T(i) = 0.5$, this meant that $\overline{f_T} = \overline{f_F}$ and therefore setting this parameter to True (or False) brought no benefit on average. If $\text{score}_T(i) < 0.5$, then $\overline{f_T} < \overline{f_F}$ and since the aim was to minimise the utility value, setting this parameter to True was beneficial. Following the same logic, score values above 0.5 had detrimental effect over the utility value. The score table provided a good overview of the amount of information contained in each variable and the likely impact of including it in the dataset on modelling accuracy.

5.2.6 Scores validation

To validate the accuracy of the parameter scores, it was only necessary to validate the model errors estimated by the cross-validation for each data collection strategy. The cost for each strategy needed no validation as it was already known and it did not change from year to year. However, for a given data collection strategy, the RMSE of each model estimated by cross-validation, as described in section 5.2.3, was likely to be different from the RMSE calculated by testing these same models against the unseen test dataset (ABR61 year 2016). A comparison of the RMSE predicted by the cross-validation over the training dataset and RMSE calculated from predictions on the test dataset, had to be made, to validate the model error estimated by the cross-validation was realistic. If the estimate of model error was

accurate it would have meant that the parameter scores were also correct, meaning that the strategy selections made by the scatter search were valid.

10000 data collection strategies were randomly sampled from the database of strategies. The absolute RMSE ($\mathrm{RMSE}_{\mathrm{abs}}$) for each model of each strategy was already recorded by the scatter search software. The RMSE for each model of the reference strategy ($\mathrm{RMSE}_{\mathrm{ref}}$) was also already known, as it was needed for calculating the relative RMSE ($\mathrm{RMSE}_{\mathrm{relative}}$). Thus, the model error for a model, trained on data reduced by a given data collection strategy, could be expressed as a ratio of the reference model, which was trained over the same data set without any reductions:

$$\mathrm{RMSE}_{\mathrm{ratio}} = \frac{\mathrm{RMSE}_{\mathrm{abs}}}{\mathrm{RMSE}_{\mathrm{ref}}}$$

The ratio was an effective way to measure model error in relation to the reference model's error. This was necessary, as the absolute value of RMSE of any model over the training data with cross-validation would have been different to the RMSE over the test dataset. For example, the RMSE of k-NN, estimated by cross-validation over the ABR61 2011-2015 dataset was 1580.8, whereas the RMSE of k-NN trained over ABR61 2011-2015 and predicting the 2016 data was 707.27. As there was a difference between the RMSE from cross-validation and the test data, it was important in both cases to express the RMSE as relative to the RMSE of the reference model tested over the same dataset.

Two ratios were calculated for each data collection strategy: $\mathrm{RMSE}_{\mathrm{train}}$ and $\mathrm{RMSE}_{\mathrm{test}}$. $\mathrm{RMSE}_{\mathrm{train}}$ was the average ratio of all machine learning models, with the errors estimated by cross-validation over the training dataset ABR61 2011-2015. For example, in the first experiment $\mathrm{RMSE}_{\mathrm{train}}$ was calculated using the following formula:

$$\mathrm{RMSE}_{\mathrm{train}} = \frac{\left(\frac{\mathrm{RMSE}_{\mathrm{knn}}}{\mathrm{RMSE}_{\mathrm{knn\ ref}}} + \frac{\mathrm{RMSE}_{\mathrm{rf}}}{\mathrm{RMSE}_{\mathrm{rf\ ref}}} + \frac{\mathrm{RMSE}_{\mathrm{gbm}}}{\mathrm{RMSE}_{\mathrm{gbm\ ref}}}\right)}{3}$$

Where:

- knn, rf and gbm were the models used to estimate the error of each strategy.
- $\mathrm{RMSE}_{\mathrm{knn}}$, $\mathrm{RMSE}_{\mathrm{rf}}$ and $\mathrm{RMSE}_{\mathrm{gbm}}$ were calculated using cross-validation over the training (ABR61 2011-2015) dataset reduced by some data collection strategy
- $\mathrm{RMSE}_{\mathrm{knn\ ref}}$, $\mathrm{RMSE}_{\mathrm{rf\ ref}}$ and $\mathrm{RMSE}_{\mathrm{gbm\ ref}}$ were the reference model RMSEs calculated using cross-validation over the same training dataset without any reductions.
- $\frac{\mathrm{RMSE}_{\mathrm{knn}}}{\mathrm{RMSE}_{\mathrm{knn\ ref}}}$ was the $\mathrm{RMSE}_{\mathrm{ratio}}$ for the knn model, and the other two fractions in the numerator were the $\mathrm{RMSE}_{\mathrm{ratio}}$ of the other two models

The formula for $\mathrm{RMSE}_{\mathrm{test}}$ was the same, but:

- The models knn, rf and gbm, used in the last equation, were trained on the whole training dataset (ABR61 2011-2015), reduced by the same data collection strategy. The reference models were trained on the same training data without any reductions.
- $\mathrm{RMSE}_{\mathrm{knn}}$, $\mathrm{RMSE}_{\mathrm{rf}}$ and $\mathrm{RMSE}_{\mathrm{gbm}}$ were calculated by predicting dry weight measurements from the test dataset (ABR61 2016). No reductions were made on the test dataset.
- $\mathrm{RMSE}_{\mathrm{knn\ ref}}$, $\mathrm{RMSE}_{\mathrm{rf\ ref}}$ and $\mathrm{RMSE}_{\mathrm{gbm\ ref}}$ were the reference model RMSEs calculated by predicting dry weight measurements of the same unreduced test dataset.

If the cross-validation predictions for RMSE were correct, then the values of $\mathrm{RMSE}_{\mathrm{train}}$ would have to be close to the corresponding $\mathrm{RMSE}_{\mathrm{test}}$. A Shapiro-Wilk test of normality (Shapiro and Wilk, 1965) was first done on the distributions of $\mathrm{RMSE}_{\mathrm{train}}$ and $\mathrm{RMSE}_{\mathrm{test}}$ to see whether it was possible to use a paired t-test to test for a difference in the two distributions. The "shapiro.test" function was used from the R programming language (R Core Team, 2017). As the two ratios were found to be non-normal, the Wilcoxon signed-rank test (Woolson, 2007) was applied instead. It tested whether the RMSE ratios were likely to have come from the same distribution.

5.2.7 Search algorithm and supporting code implementation

The scatter search algorithm described in this section, along with the procedures for loading and organising data, calculating data collection strategy costs, modelling error, building

custom machine learning models and the parameter scores validation, were all implemented in the Python 2.7 (Python Software Foundation, 2013) programming language. Implementations of the naïve and GA models were also written in Python. A fundamental part of the model related procedures and the naïve and GA model implementations was the code used for training machine learning models, which was written in R (R Core Team, 2017), and made use of several R packages implementing machine learning algorithms: the base package provided linear regression (R Core Team, 2017), the randomForest package implemented random forest (Liaw and Wiener, 2002), k-nearest neighbour was done using the kknn package (Schliep and Hechenbichler, 2016), the gbm package provided generalised boosted models (Ridgeway, 2015), and SVM (part of the GA model) were trained using caret and liblinear packages (Kuhn, 2016; Helleputte, 2017).

5.3 Results

5.3.1 Simple machine learning model experiment

Minimum and maximum RMSE calculation

The first step to successfully running the scatter search method was finding the minimum and maximum RMSE values ($\mathrm{RMSE}_{\mathrm{min}}$ and $\mathrm{RMSE}_{\mathrm{max}}$), used for scaling most of the absolute RMSE values ($\mathrm{RMSE}_{\mathrm{absolute}}$) between 0 and 1. Table 5.6 lists the reference model error of each model over the full training dataset (ABR61, years 2011-2015), calculated through cross-validation. Descriptive statistics for that dataset can be found in Table 5.7. As described in section 5.2.3, for the initial run of the scatter search method the minimum and maximum RMSE values were set to: $\mathrm{RMSE}_{\mathrm{min}} = \mathrm{RMSE}_{\mathrm{ref}}$ and $\mathrm{RMSE}_{\mathrm{max}} = 2 \times \mathrm{RMSE}_{\mathrm{ref}}$, as the reference RMSE was expected to be the lowest possible. The minimum and maximum RMSE values from the initial run are shown in Table 5.8.

The initial run generated 20000 strategies, which allowed a visual inspection of the RMSE distribution for each model, shown on Figure 5.4. It was evident that the initial procedure used for calculating the minimum ($\mathrm{RMSE}_{\mathrm{min}} = \mathrm{RMSE}_{\mathrm{ref}}$) and maximum ($\mathrm{RMSE}_{\mathrm{max}} = 2 \times \mathrm{RMSE}_{\mathrm{ref}}$) error was inadequate. This led to the updated procedure which generated better borders, seen on Figure 5.5.

The aim was to capture the main part of the distribution, where most of the strategies' errors lied, while at the same time imposing higher penalties on strategies with outlying errors above

the maximum. Setting the maximum RMSE below the outliers meant that they would have a $\mathrm{RMSE}_{\mathrm{relative}}$ above 1, which increased selection against these strategies. The minimum RMSE was set so there would be small distance between it and the smallest observed RMSE, which left some space for solutions with smaller model errors. While it was not crucial for all $\mathrm{RMSE}_{\mathrm{relative}}$ to be scaled between 0 and 1, it allowed for finer control over the scatter search algorithm strategy selection, like for example the increase of modelling accuracy importance over data collection cost which was achieved via the 0.1 scaling factor in the utility function.

	k-NN	Random forests	GBM
$RMSE_{ref}$	1580.8	1458.6	1733.3

Table 5.6 Model error of reference models calculated using cross-validation over the ABR61 dataset, years 2011-2015. The data used for training the reference models was not reduced by any data collection strategy. Note: the values presented in this table have been rounded to the nearest tenth.

Genotype	Mean	Standard deviation	Minimum value	25%	50%	75%	Maximum value
EMI-11	1103	874	17	449	904	1511	3115
Giganteus	1489	1357	3	493	1022	2222	5798
Goliath	1269	897	6	577	1273	1677	3446
Sac-5	1411	1239	1	528	1092	2105	4527

Table 5.7 Descriptive statistics for the dry weight values in the training data (years 2011 to 2015 from the ABR61 dataset).

	k-NN	Random forests	GBM
$RMSE_{min}$	1580.8	1458.6	1733.3
$RMSE_{max}$	3161.5	2917.1	3466.7

Table 5.8 Minimum and maximum RMSE values for the first run of the scatter search algorithm. These values were used for scaling most of the $RMSE_{abs}$ for each model to values between 0 and 1.

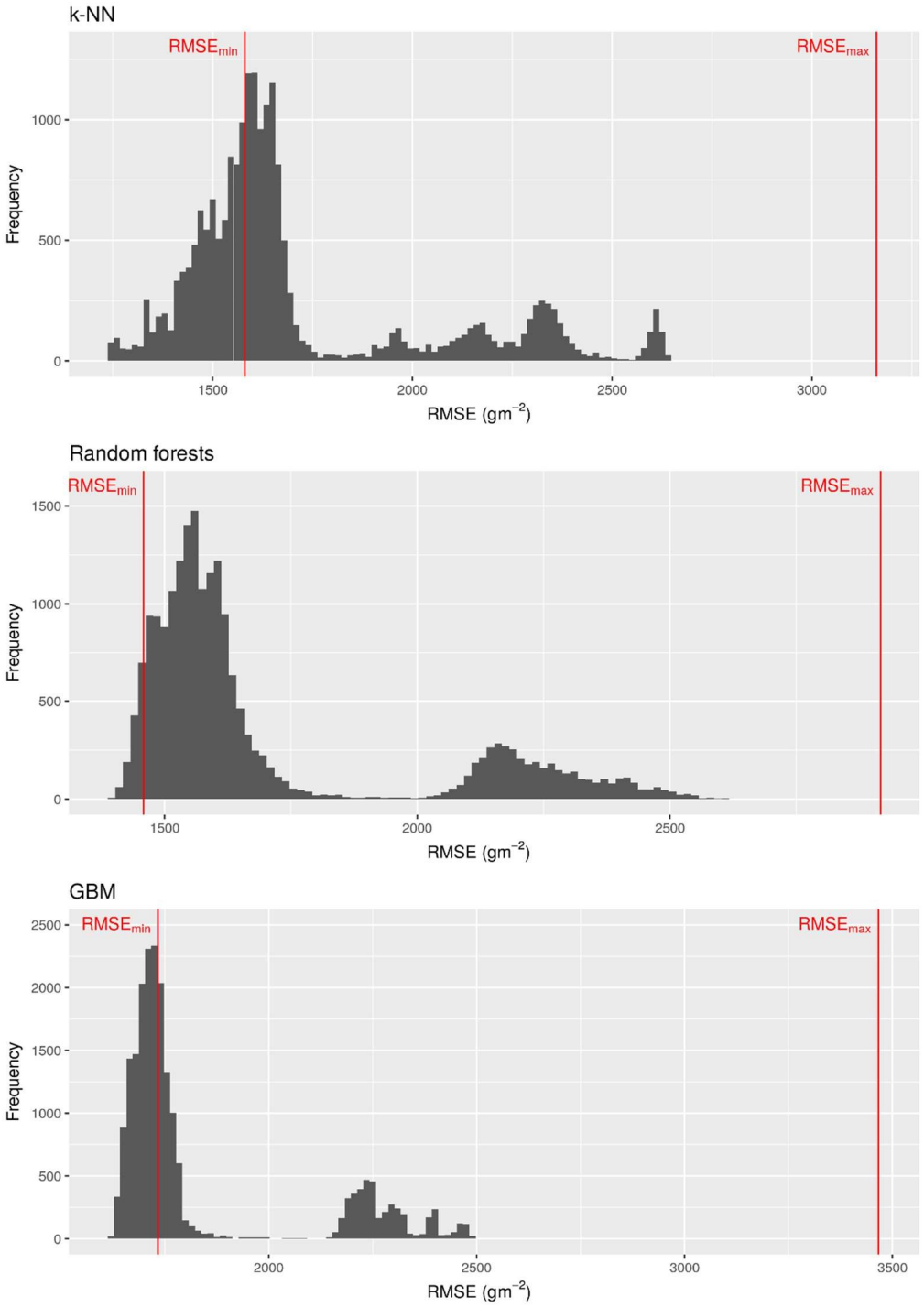

Figure 5.4 Histogram of the RMSE values of each model taken from the 20000 data collection strategies generated by the initial run of the scatter search method. The minimum and maximum RMSE parameters, used for scaling the absolute RMSE of each model, are shown as red lines. They were used for the initial run of the scatter search method, which generated the 20000 strategies. $RMSE_{min}$ was set to be equal to the reference RMSE. As the maximum was far above the highest error and there were many strategies with error below the minimum, these parameters were considered inadequate.

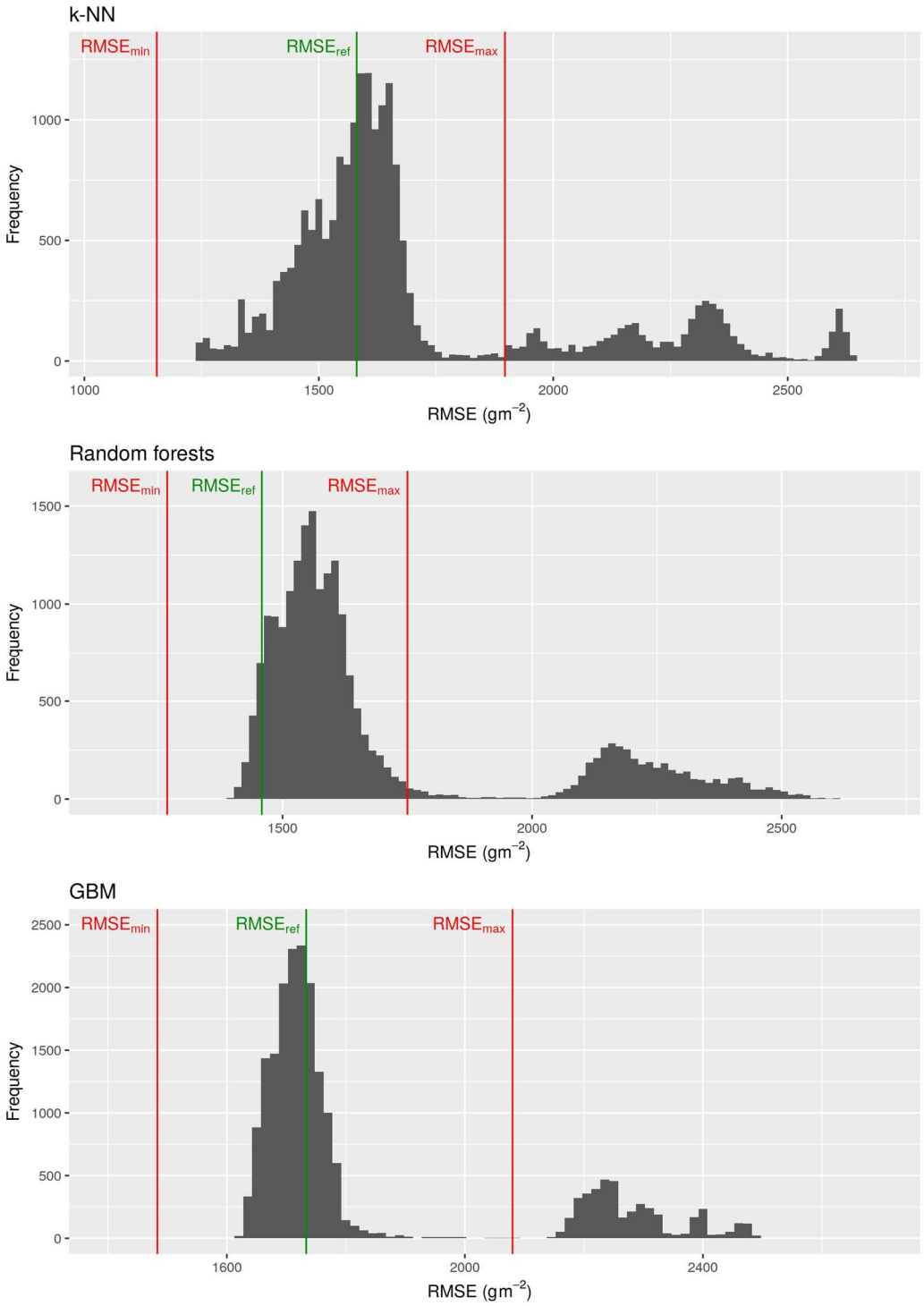

Figure 5.5 Histogram of the RMSE values of each model taken from the 20000 data collection strategies generated by the initial run of the scatter search method. The $RMSE_{min}$ and $RMSE_{max}$ parameters used for the second run are shown as red lines. The green line is the reference RMSE. This figure demonstrates the effect of the changes to the calculation of the minimum and maximum parameters. The aim was to include the main part of the RMSE distribution (most strategies), between the minimum and maximum and to impose higher penalty on the outliers above the maximum. The minimum and maximum values shown here were lower compared to the ones on Figure 5.4. This had the effect placing strategies with lower RMSE than the reference above the minimum and placing outlier strategies with high RMSE above the maximum. This prevented low RMSE strategies from having a negative relative RMSE and it made high RMSE strategies (those above the maximum) less likely to be considered good.

Best strategy and variable scores

The updated $\mathrm{RMSE}_{\min}$ and $\mathrm{RMSE}_{\max}$ were used in a second run of the scatter search method. The strategies database was emptied from the 20000 strategies, generated in the previous run, as it was assumed that the selection decisions made by the algorithm would have been different after modifying the minimum and maximum parameters. The method was run for 252 iterations which created a database of 27660 unique data collection strategies. The score table was calculated over the whole database, and it is shown on Figure 5.6. The values shown in the figure are the $\mathrm{score}_{\mathrm{T}}$ for each parameter. The ones with scores lower than 0.5 were beneficial as they minimised the utility value on average, while the ones above 0.5 were detrimental as they increased it. The values from the figure are also listed in Table 5.9.

The high score of the month of May ($\mathrm{score}_{\mathrm{T}}(\mathrm{May}) = 0.593$) meant that it increased the utility value overall, which could have been due to this time being too early in the season for useful measurements. The plants usually emerged in April, and during May they had grown by a small amount. May had the highest score among all parameters, putting it in the first place for detrimental effect over the utility value.

The month of June is missing from the score table, as there were no dry weight measurements done during that month in the ABR61 dataset, years 2011-2015. As the simple machine learning models required records with dry weight measurements, data collected in June could not be used with that experiment. The month of July had the lowest score among all parameters, which suggests that it contained very important data points. The most vigorous plant growth was observed during July, but a plateau in that growth was reached in August.

No significant importance of meteorological variables was observed – but this most certainly does not mean that they were not important. ABR61 was a single location trial, so all plots experienced the same weather conditions. If this dataset was comprised of multiple locations, it is likely that the results will show the meteorological variables as more important.

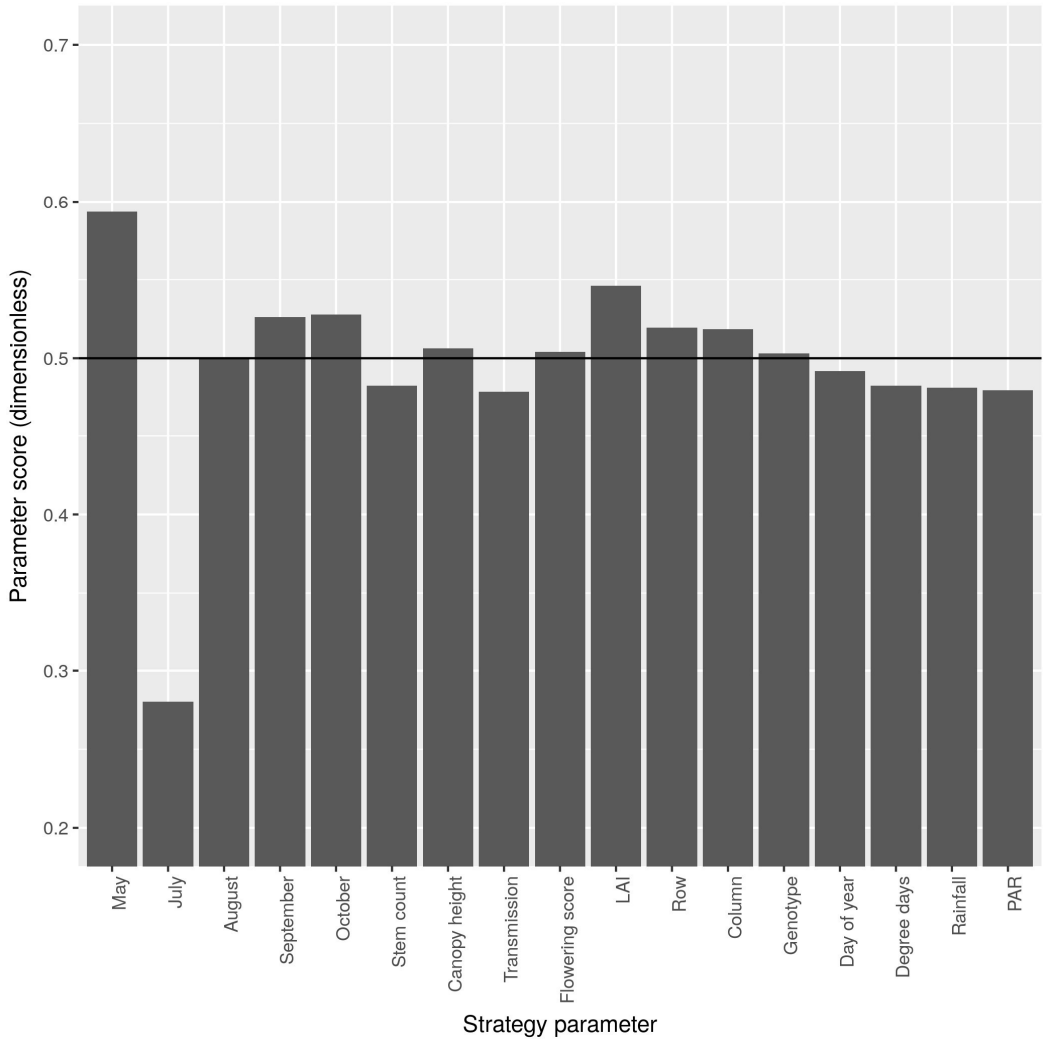

Figure 5.6 Score values ($score_T$) for each strategy parameter in the simple machine learning model experiment. Parameters below the 0.5 line were on average beneficial to add to the data collection scenario, as they helped minimise the utility value. Parameters with scores above 0.5 were detrimental as they increased the utility value.

Name	Score
May	0.593
July	0.280
August	0.500
September	0.526
October	0.527
Stem count	0.482
Canopy height	0.506
Transmission	0.479
Flowering score	0.504
Leaf area index (LAI)	0.546
Row	0.519
Column	0.518
Genotype	0.503
Day of year	0.491
Degree days	0.482
Rainfall	0.481
Photosynthetically active radiation (PAR)	0.479

Table 5.9 Score values for each strategy parameter from the simple machine learning experiment.

This could have been the reason for the lower scores for July, as well as the higher scores in September and October: during these months senescence occurred in some genotypes, causing a reduction in the LAI, without reducing the biomass yield significantly. The reduction of green LAI in later months also caused additional issues with the transmission measurements, which were not as reliable as in the previous months. Transmission measured the amount of light not absorbed by the leaves, but for this measurement to be precise, the light had to be absorbed by a green leaf, as it was then used for photosynthesis. If some of the light was absorbed by senesced leaves, then that changed what the transmission measurement meant.

When designing a data collection strategy using the score values, the reference set, the set of data collection strategies the scatter search algorithm tried to improve, also needs to be considered. The reference set contained the best solutions encountered by the scatter search algorithm, and so provide a practical example of which strategy parameters were used and what effect this had. The strategies from the reference set, as they were in the 252nd iteration, are shown on Figure 5.7.

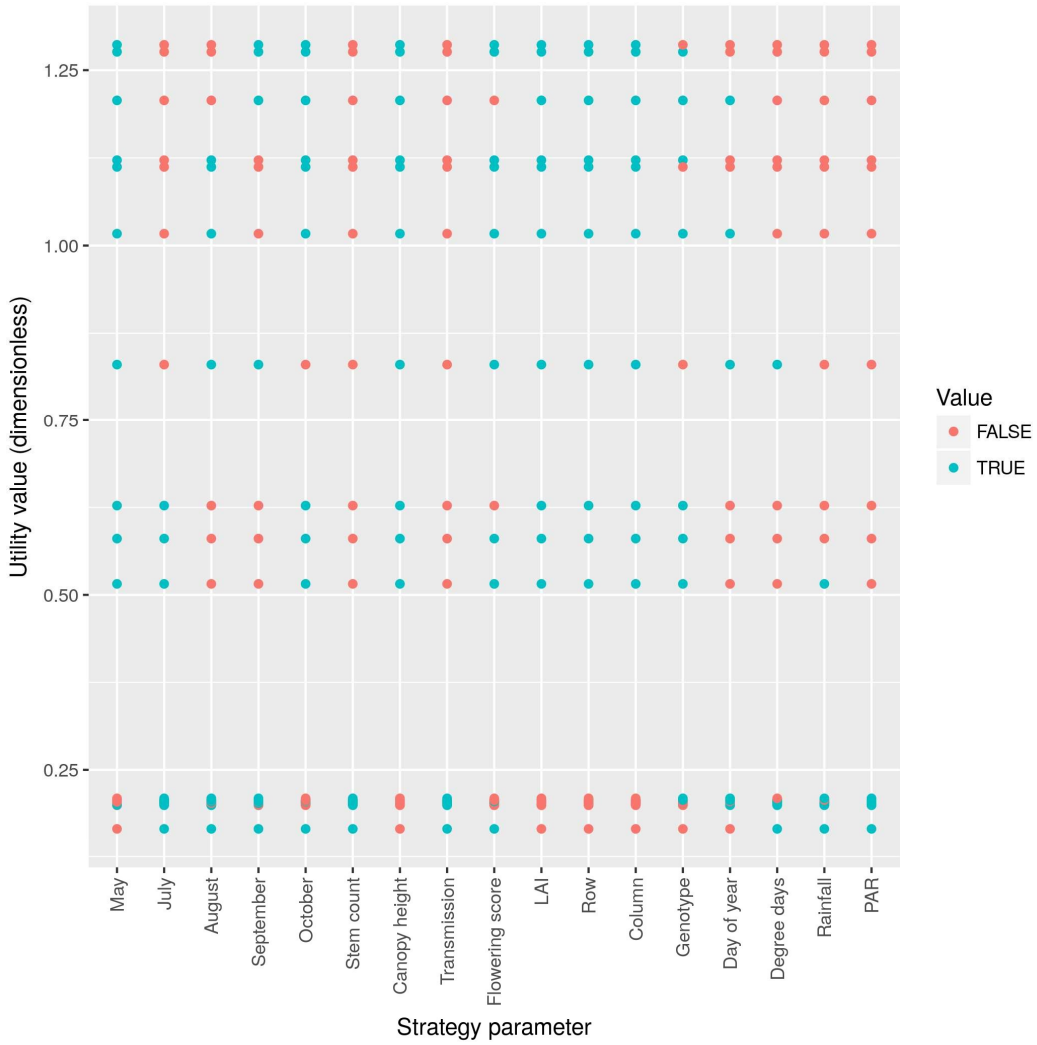

Figure 5.7 Reference set from iteration 252 of the scatter search algorithm in the simple machine learning experiment. Each horizontal band of points was a single data collection strategy. The number of strategies shown on the figure is 20, as this was the full content of the reference set. The utility value of each strategy is shown on the y-axis, with the best strategy plotted at the bottom. The cluster of strategies just under the 0.25 mark were very close in utility value, so they were plotted on top of each other. The colour of each point shows whether each parameter was used in the corresponding strategy or not. Both possible values for each parameter were represented in the data collection strategies shown here, which was a result of the scatter search's approach for maintaining diversity in the reference set. Day of emergence was used in all strategies implicitly – the meteorological variables were cumulative values since emergence.

The y-axis shows the utility value for each of the strategies, with the lowest strategy being the best in the set, and the colour of the points denotes whether a variable/month was used or not. Each horizontal band of points in the figure denotes one data collection strategy, although the points of some data collection strategies are overlapping, because their utility values were too close to each other. The months September and October were used in the best strategy, despite having a score higher than 0.5. This suggests that there may be interaction between the parameters, which cannot be detected by examining the score values alone.

The month scores demonstrated that the impact the caveat of the cost function had on the strategy utility value, and subsequently on the score for each month, was in fact minimal. This can be seen from the higher score of May, compared to September and October, despite the last two months having significantly fewer measurements as shown on Figure 5.2.

The parameter with the second highest score was LAI. The reason for this was twofold: the cost of collecting LAI was significantly higher compared to the rest of the variables. Also, as can be seen on Figures 3.11, 3.12, and 3.15, LAI was a noisy measurement, which plateaued and declined towards the end of the growing season. Its changing relationship with dry matter yield made modelling using LAI as a predictor a difficult task.

Variables with scores below 0.5 that looked promising included stem count, transmission, and the meteorological variables degree days, rainfall and PAR. All these variables were included in the best strategy. The accuracy of the parameter scores became more uncertain with scores closer to 0.5. These parameters sometimes had a small positive or small negative effect on the model RMSE, possibly caused by interactions with another present or missing parameter. For example, the best strategy did not use day of year, although its score was below 0.5, and used flowering score, which had a score higher than 0.5. This once again demonstrated the need to consider the context in which a parameter was used, i.e. which other parameters were also set to True.

There was also a small variability in RMSE associated with training some models, in randomisation-based models like random forests and GBM. For example, if multiple random forests models were trained on the same dataset, each one would have had a slightly different RMSE. This variability in the RMSE was caused by the randomness in the training which did

not produce the same model each time, despite being trained on the same dataset. This meant that the RMSE calculated by cross-validation had some variability associated with it and the only solution was to cross-validate the same model multiple times and take the mean RMSE. However, this was computationally infeasible as it would have increased the computation time dramatically for very small gains in RMSE accuracy. Thus, cross-validation models were trained and tested only once each fold.

The best data collection strategy found in this experiment, illustrated in Figure 5.7, was used in some further work, outside of the scope of this chapter, done to illustrate the effect of this strategy on model accuracy. Each of the three machine learning algorithms (RF, KNN and GBM) was used to train reference models, on the whole training dataset ABR61, years 2011-2015. The dataset was reduced by the best data collection strategy, and the three algorithms were used again to train best strategy models on the reduced dataset. All models made dry weight predictions on the unseen test dataset (ABR61, year 2016), and their predictions were plotted against the real data. Figure 5.8 shows real against predicted dry weight values in each model. Figure 5.9 shows the real and predicted dry weight values in the random forests models, Figure 5.10 shows the same for k-nearest neighbour models, and Figure 5.11 shows the same in GBM models.

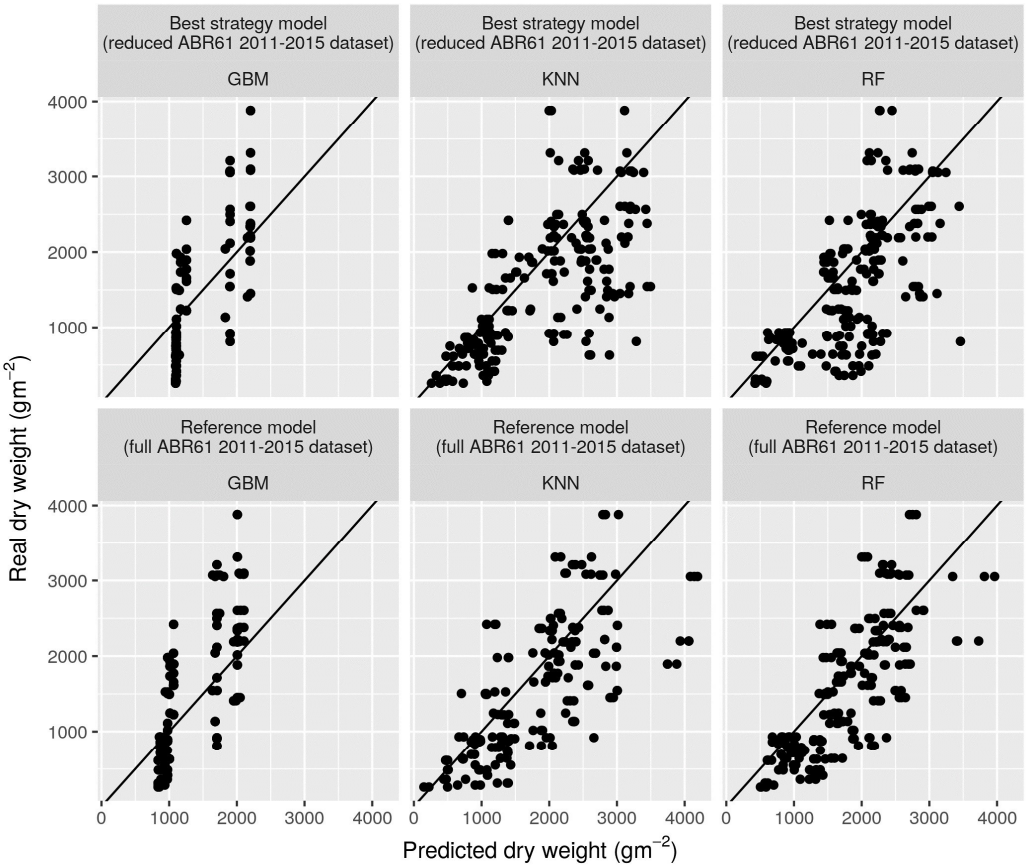

Figure 5.8 Real vs predicted dry weight for the ABR61 dataset, year 2016. Each point was the dry weight for a single plant. The models that made the predictions were part of the scatter search method described in chapter 5. The three machine learning algorithms (KNN, RF and GBM) and the best data collection strategy were used to train reference and best strategy models – the reference models were trained on the full ABR61 dataset, whereas the best strategy models were trained on the same dataset reduced by the best data collection strategy. This resulted in three best strategy and three reference models, which predicted dry weight in the test ABR61 2016 dataset. The test dataset was not given to the models during training.

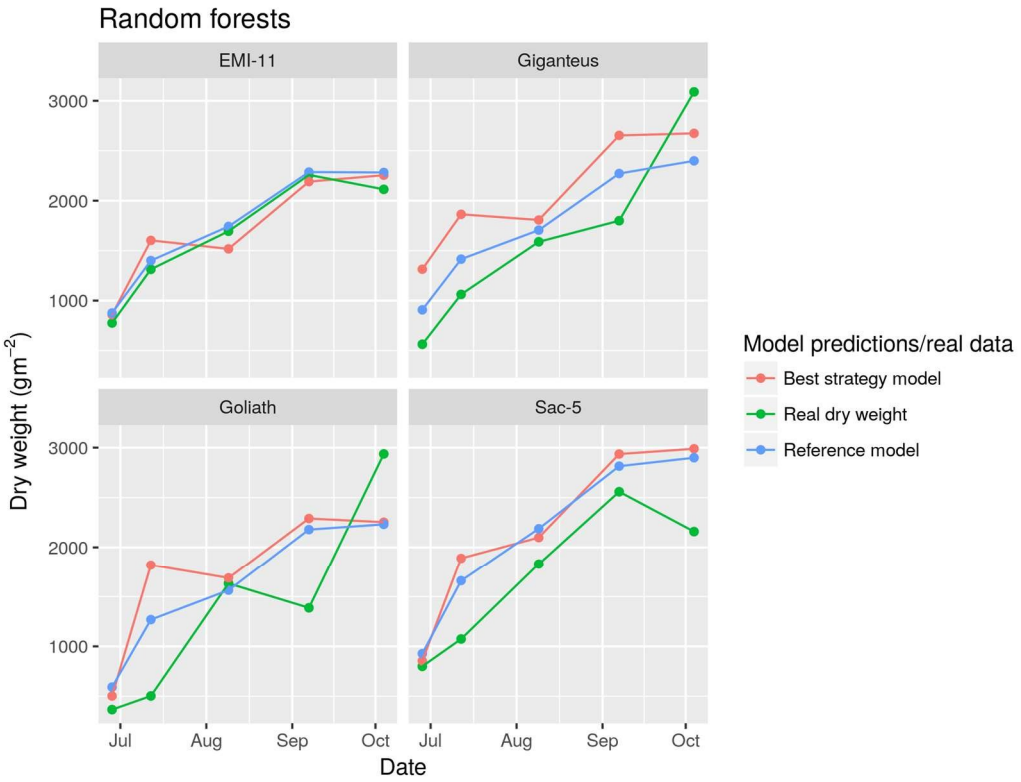

Figure 5.9 Predicted and actual values of dry weight in the ABR61 dataset, year 2016. The predicted dry weight for each plant were averaged by plot, and then averaged again by genotype. The real dry weight (which was a plot-level measurement), was averaged by genotype. The two models used to make the predictions were trained using random forests. The reference model was trained on the ABR61 dataset, years 2011-2015. The best strategy model was trained on the same dataset, reduced by the most optimal data collection strategy. The two models predicted dry weight in year 2016 of the ABR61 trial, plotted here. The real dry weight values are also shown in green.

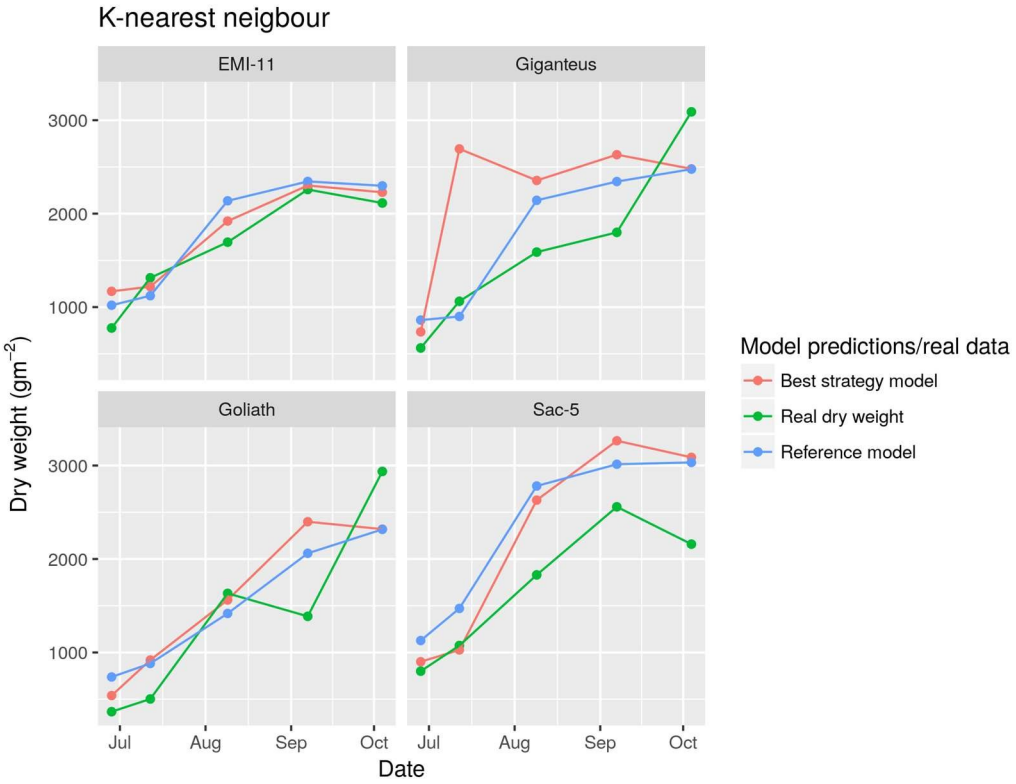

Figure 5.10 Predicted and actual values of dry weight in the ABR61 dataset, year 2016. The real and predicted dry weight values were as described in Figure 5.9. The two models used to make the predictions were trained using k-nearest neighbour. The reference model was trained on the ABR61 dataset, years 2011-2015. The best strategy model was trained on the same dataset, reduced by the most optimal data collection strategy. The two models predicted dry weight in year 2016 of the ABR61 trial, plotted here. The real dry weight values are also shown in green.

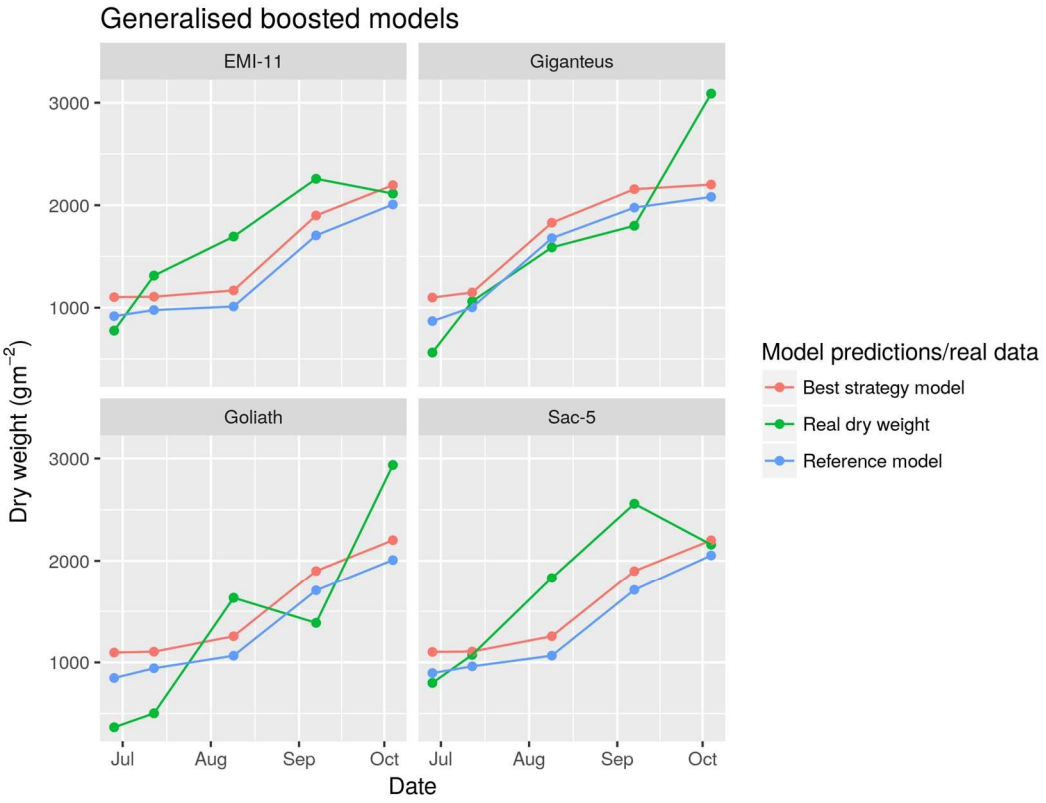

Figure 5.11 Predicted and actual values of dry weight in the ABR61 dataset, year 2016. The real and predicted dry weight values were as described in Figure 5.9. The two models used to make the predictions were trained using generalised boosted models. The reference model was trained on the ABR61 dataset, years 2011-2015. The best strategy model was trained on the same dataset, reduced by the most optimal data collection strategy. The two models predicted dry weight in year 2016 of the ABR61 trial, plotted here. The real dry weight values are also shown in green.

Validation

The Shapiro-Wilk test showed that both ratios ($\mathrm{RMSE}_{\mathrm{train}}$ and $\mathrm{RMSE}_{\mathrm{test}}$) were not normally distributed ($\mathrm{p} < 2.2 \times 10^{-16}$). As this violated one of the t-test assumptions, it could not be applied to comparing the two distributions, and instead the Wilcoxon signed-rank test was used. The test showed that the $\mathrm{RMSE}_{\mathrm{train}}$ and $\mathrm{RMSE}_{\mathrm{test}}$ ratios were not drawn from the same distribution ($\mathrm{p} < 2.2 \times 10^{-16}$) and a one-sided Wilcoxon signed-rank test showed that the median of $\mathrm{RMSE}_{\mathrm{train}}$ was greater than that of $\mathrm{RMSE}_{\mathrm{test}}$. The test also calculated the pseudomedian - the median of the difference between a sample from the $\mathrm{RMSE}_{\mathrm{train}}$ and $\mathrm{RMSE}_{\mathrm{test}}$ distribution, which was 0.026. The median, mean and standard deviation for the distribution of the ratio differences ($\mathrm{RMSE}_{\mathrm{train}} - \mathrm{RMSE}_{\mathrm{test}}$) are listed in Table 5.10, and a histogram of the distribution is shown on Figure 5.12.

Median	Mean	SD	N
0.022	0.06	0.17	10000

Table 5.10 Statistics for the ratio difference distribution ($RMSE_{train} - RMSE_{test}$). Calculated using the "median", "mean" and "sd" functions in R (R Core Team, 2017).

The two ratios were paired according to the data collection strategy they related to. A positive ratio difference meant that cross-validation overestimated the RMSE of the model, compared to the RMSE over the test dataset. The median and mean were small positive numbers which meant that cross-validation overestimated the model RMSE slightly. However, there were cases where the overestimation was significant, as seen by the outliers above 0.25 on the histogram.

The raw RMSE values of the 10000 data collection strategy samples can be seen in section 8.2 in the appendix. Figure 8.1 shows the train RMSE values, obtained from cross-validating each model on the ARB61 2011-2015 dataset. Figure 8.2 shows the test RMSE values, which were calculated from dry weight predictions that these models made on the unseen ABR61 test dataset, year 2016, as well as real dry weight data.

5.3.2 Compound models experiment

Minimum and maximum RMSE calculation

The selection of minimum and maximum RMSE followed the same logic as in the previous experiment. First, the reference RMSE was calculated for the two models (naïve and GA model) by performing cross-validation over the ABR61 dataset, years 2011-2015. The results

are shown in Table 5.11. The $\mathrm{RMSE}_{\mathrm{min}}$ and $\mathrm{RMSE}_{\mathrm{max}}$ for the initial run were: $\mathrm{RMSE}_{\mathrm{min}} = \mathrm{RMSE}_{\mathrm{ref}}$ and $\mathrm{RMSE}_{\mathrm{max}} = 2 \times \mathrm{RMSE}_{\mathrm{ref}}$, i.e. the initial guess for what the optimal minimum and maximum might be. This increased selection against strategies with RMSE higher than twice as much as $\mathrm{RMSE}_{\mathrm{ref}}$. The scatter search procedure was run with these values, and generated 6966 strategies in the database. This number was a lot lower than the 20000 strategies from the previous experiment, but the algorithm was stopped as performing cross-validation for each strategy took considerably longer time. This was because the naïve and GA models took longer to train compared to the simple machine learning models.

A histogram of the initial 6966 strategies is shown on Figure 5.13. As was the case for the previous experiment, the initial $\mathrm{RMSE}_{\mathrm{min}}$ and $\mathrm{RMSE}_{\mathrm{max}}$ were inadequate, considering that a lot of strategies' RMSE fell outside these boundaries, and the maximum boundary was set too high. The improved method for calculating $\mathrm{RMSE}_{\mathrm{min}}$ and $\mathrm{RMSE}_{\mathrm{max}}$ produced the minimums and maximums shown on Figure 5.14. This ensured that all data collection strategies' RMSE were above the $\mathrm{RMSE}_{\mathrm{min}}$ and those with high RMSE values that were outliers from the main distribution would be selected against. The high RMSE outliers are seen on the naïve model RMSE histogram.

	Naïve model	GA model
$RMSE_{ref}$	1861.1	1988.9

Table 5.11 Model error of reference models calculated using cross-validation over the ABR61 dataset, years 2011-2015. The data used for training the reference models were not reduced by any data collection strategy. Note: the values presented in this table have been rounded to the nearest tenth. Descriptive statistics for the dataset are shown in Table 5.7.

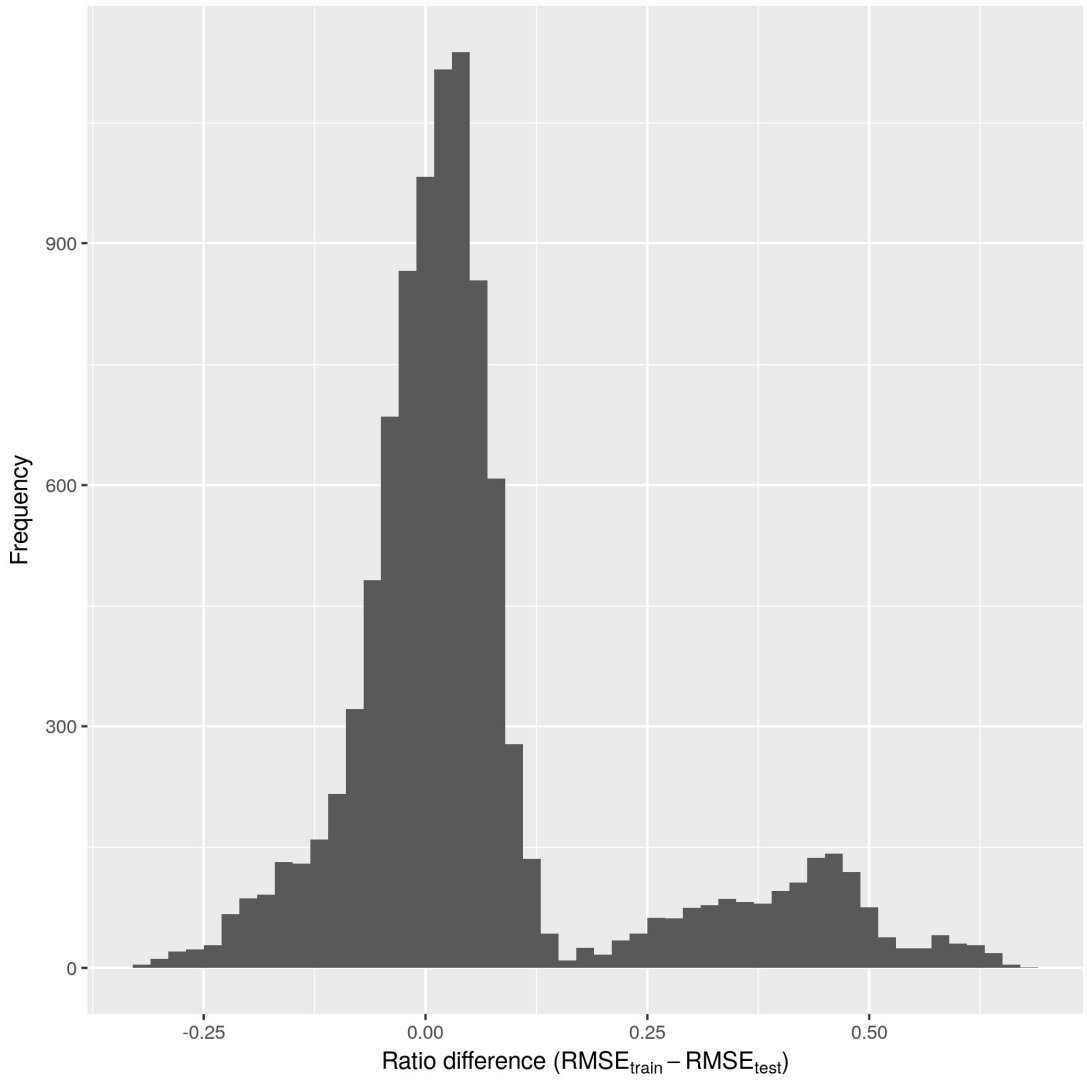

Figure 5.12 Histogram of the differences between RMSE ratios for each of the 10000 data collection strategy samples from the database. The RMSE for a model was overestimated if the ratio difference was above 0, and was underestimated if it was below 0.

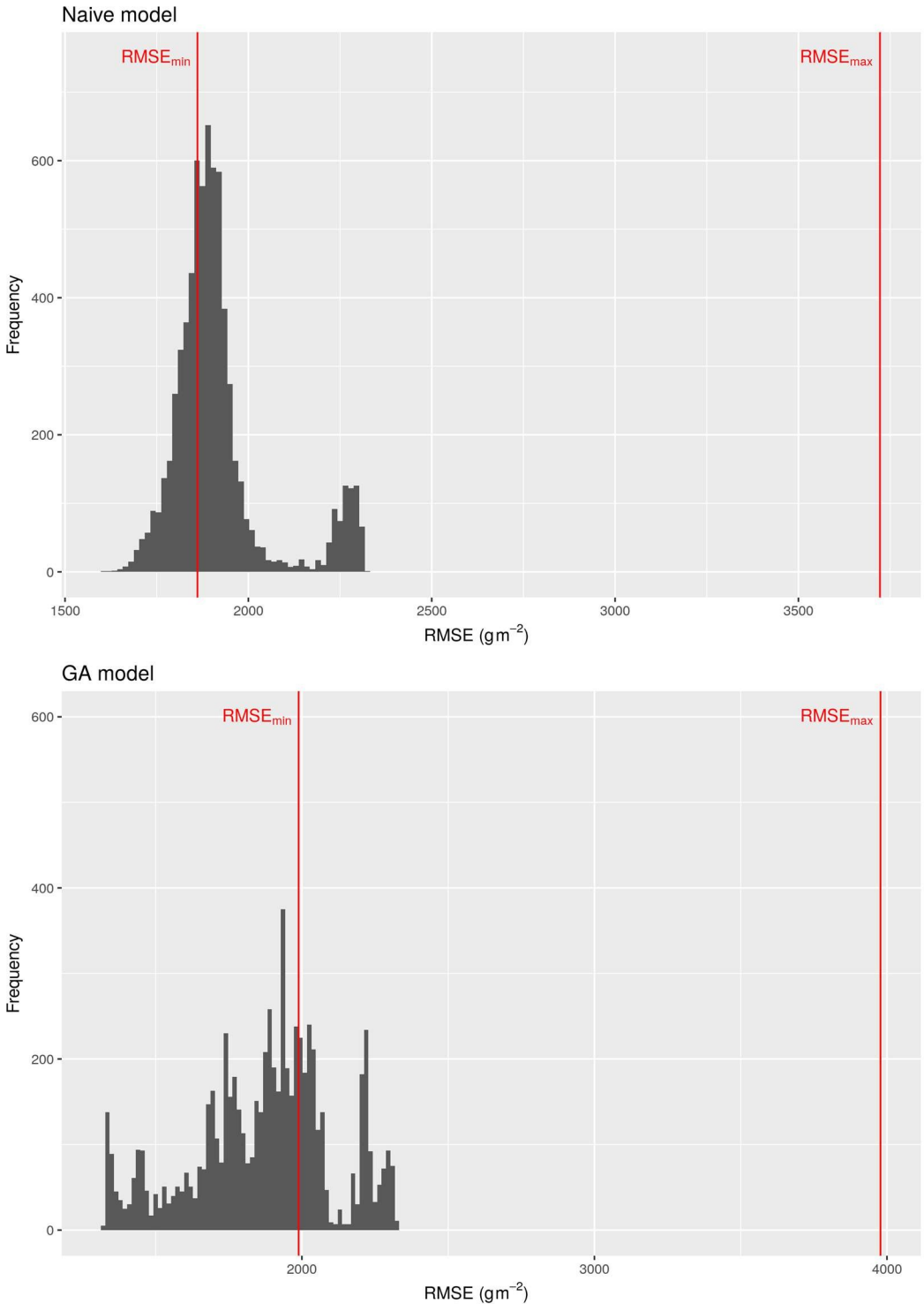

Figure 5.13 Histogram of the RMSE of the naïve and GA models across all 6966 strategies generated from the initial run of the scatter search method. The minimum and maximum RMSE parameters, used for scaling the absolute RMSE of each model in the initial run, are shown as red lines. The minimum RMSE was set to be the same as the reference RMSE. As the maximum was far above the highest error and there were many strategies with error below the minimum, these parameters were considered inadequate.

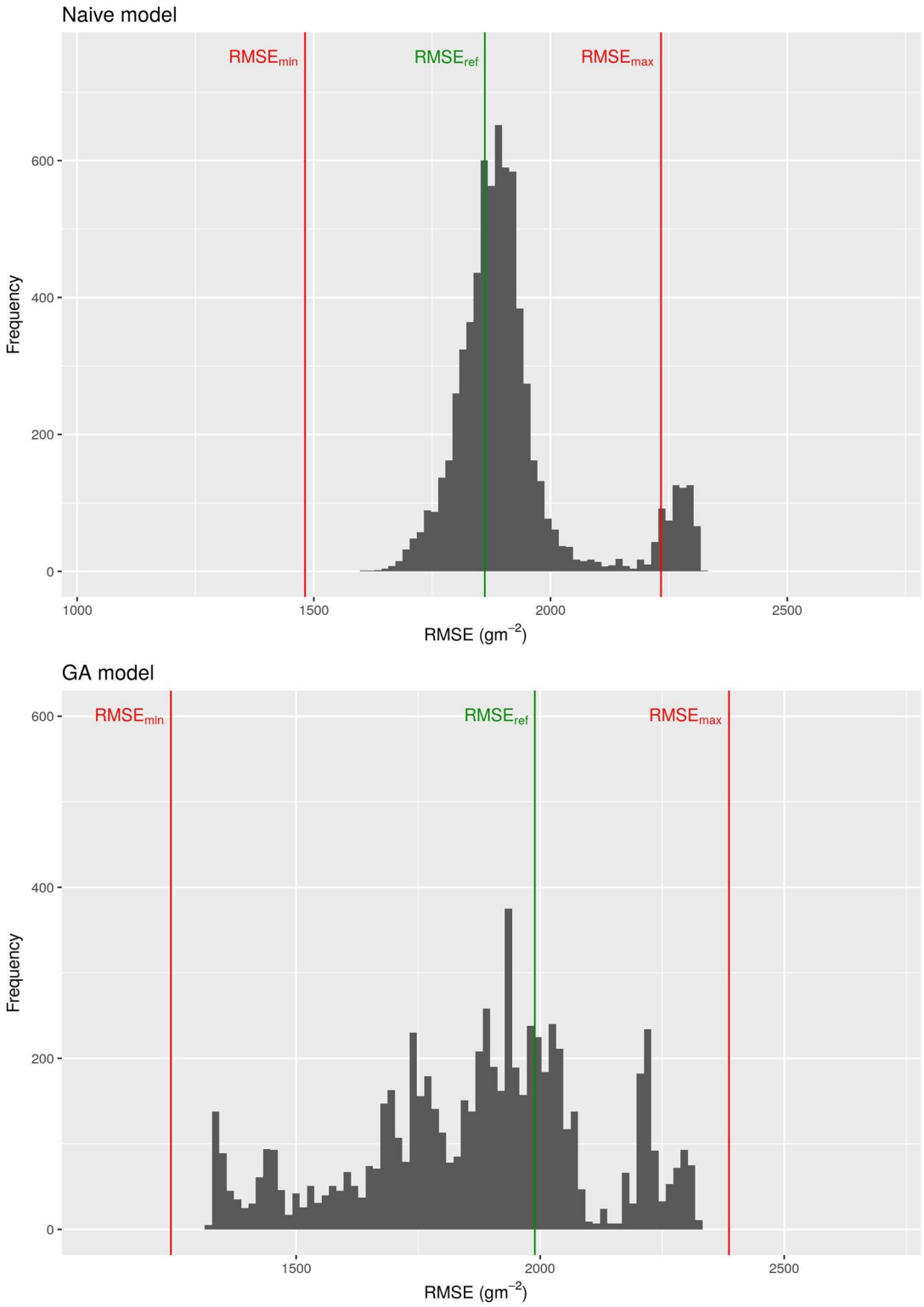

Figure 5.14 Histogram of the RMSE of the naïve and GA models for all 6966 strategies generated by the initial run of the scatter search algorithm. $RMSE_{min}$ and $RMSE_{max}$ values calculated using the improved procedure for calculating minimum and maximum RMSE, are shown as red lines. The reference RMSE is shown as a green line. This figure demonstrates the effect of the changes to the calculation of the minimum and maximum parameters. The aim was to include the main part of the RMSE distribution (most strategies), between the minimum and maximum and to impose higher penalty on the outliers above the maximum. The minimum is now below the smallest RMSE for both models. The maximum for the naïve model was set just below a cluster of high RMSE outlier strategies, which lowered their chances of being considered good by the algorithm.

Best strategy and variable scores

The second run of the scatter search method completed 295 iterations. Despite the higher number of iterations compared to the previous experiment, the strategies in the database were 17773, whereas in the simple machine learning experiment, the method generated 27660 strategies in 252 iterations. This was caused by the smaller size of the reference set ($b = 8$ for this experiment compared to $b = 20$ for the previous one).

The scores for each strategy parameter are listed under Table 5.12 and are shown in Figure 5.15. As in the previous experiment, the month of July had the lowest score. The months August, September and October were very close to 0.5, which suggested that collecting data during that time would not impact model accuracy badly. The high scores of May and June, suggested that this time was too early for taking meaningful measurements, which agreed with the results from the simple machine learning model experiment.

The month scores of this experiment again demonstrated that the issue with the cost function was minimal. June and July had very different scores, despite having almost the same number of measurements as shown on Figure 5.2.

LAI had the highest score among the variables, as it did in the results of the previous experiment. This confirmed that the information gain from LAI did not justify the cost for collecting it. Other variables had switched their place with relation to the 0.5 line in this experiment. Flowering score, day of year and rainfall were below the line in the previous experiment and were above the line in this one. Genotype was above the line in the previous experiment and was below in this one.

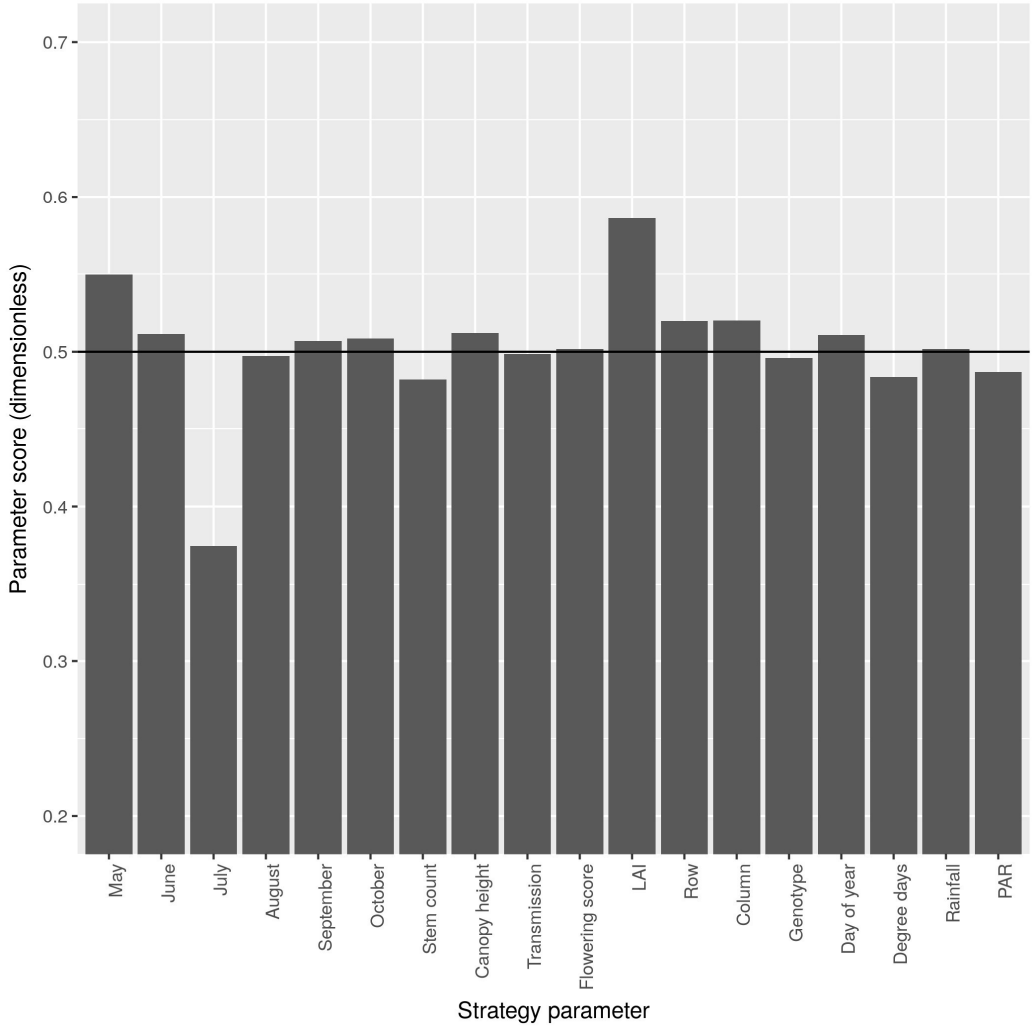

Figure 5.15 Score values ($score_T$) for each strategy parameter in the compound model experiment. Parameters below the 0.5 line were on average beneficial to add to the data collection scenario, as they helped minimise the utility value. Parameters with scores above 0.5 were detrimental as they increased the utility value.

Name	Score
May	0.549
June	0.511
July	0.374
August	0.497
September	0.506
October	0.508
Stem count	0.481
Canopy height	0.512
Transmission	0.498
Flowering score	0.501
Leaf area index (LAI)	0.585
Row	0.519
Column	0.520
Genotype	0.496
Day of year	0.510
Degree days	0.483
Rainfall	0.502
PAR	0.487

Table 5.12 Score values for each strategy parameter from the compound model experiment.

The scores for these and the other variables, apart from LAI, were very close to 0.5, so their position with respect to the 0.5 line should not be the deciding factor for including them in the measurements or not. The 0.5 line should not be taken as a hard border, but rather the distance from it should be an indicator of how significant the benefit is from including or excluding a given variable.

This idea was evident from the group of four best strategies from the reference set, shown on Figure 5.16. While some of them used parameters that were above but close to the 0.5 line, they still produced accurate models and achieved low utility scores. It is probable that this was caused by interactions between the parameters, as suggested before. Identifying these interactions was beyond the scope of the scatter search method.

Beside the issue with the uncertainty of parameter scores, another possible cause for the discrepancies between the results of this and the previous experiment were the different sets of models being evaluated in the two experiments: k-NN, random forests and GBM in the first and naive and GA model in the second.

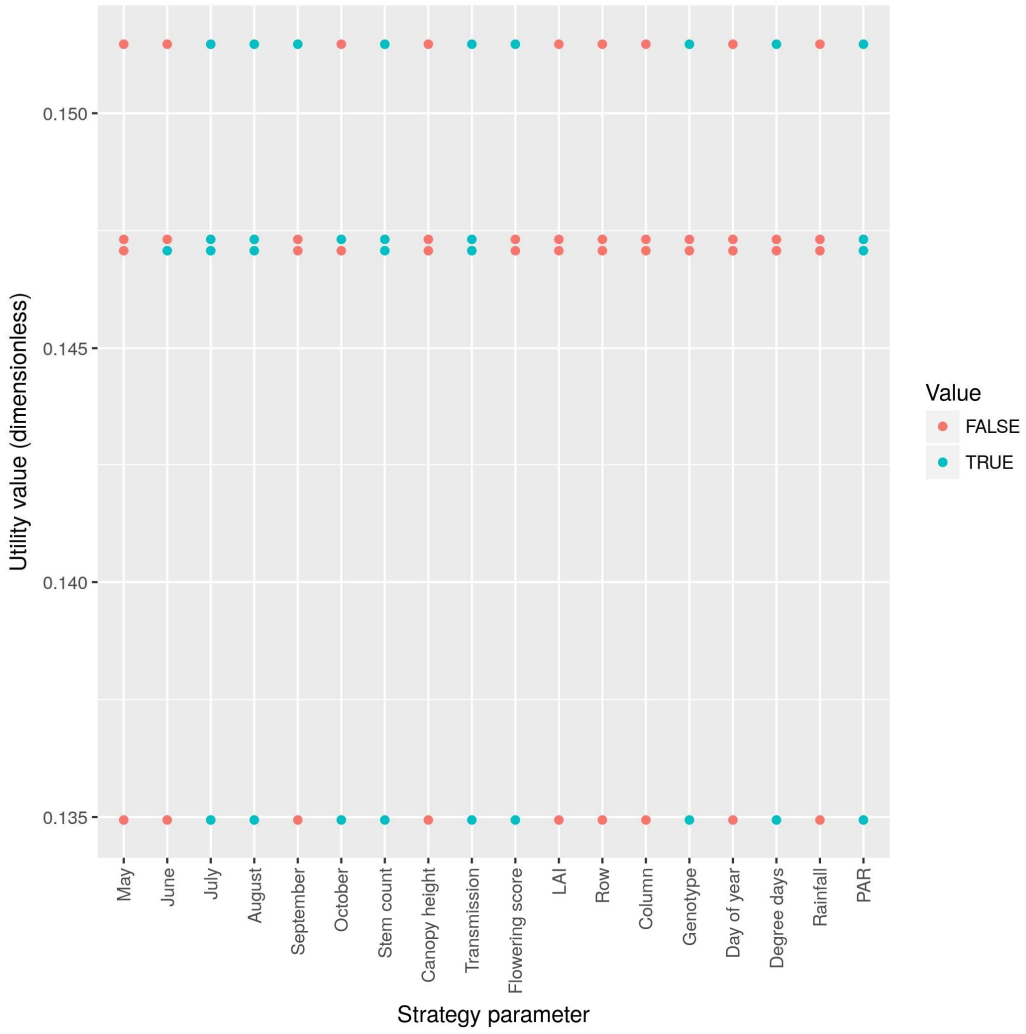

Figure 5.16 The best 4 strategies from the reference set from iteration 295 of the scatter search algorithm in the compound model experiment. The utility value of each strategy is shown on the y-axis, with the best strategy plotted at the bottom. The colour of each point shows whether each parameter was used in the corresponding strategy or not. The strategies' utility values were very close to each other, and the plot demonstrated that strategies using parameters with scores slightly higher than 0.5 could still be the found to be optimal.

The different approach these models took to modelling the data was expected to influence the results from each experiment – some parameters could have been more important in one set of models than the other. Another source of discrepancy was the differences in the training data available to the two sets of models – the simple machine learning models could only use records with dry weight measurements, which were only a subset of the full ABR61 2011-2015 dataset. Data records without measurements for dry weight were only available to the compound models, which was expected to cause further differences in the results.

Using a similar approach as the one described in the previous experiment, the best data collection strategy, illustrated in Figure 5.16, was used in further work to illustrate its effect on model accuracy. Each of the two compound models (naïve and GA) was trained as a reference model, on the whole training dataset ABR61, years 2011-2015. The dataset was reduced by the best data collection strategy, and the two models trained on the reduced dataset. The resulting models from the reduced dataset training were called best strategy models. All models made dry weight predictions on the unseen test dataset (ABR61, year 2016), and their predictions were plotted against the real data. Figure 5.17 shows real against predicted dry weight values in each model. Figure 5.18 shows the real and predicted dry weight values in the naïve models and Figure 5.19 shows the same for the GA models.

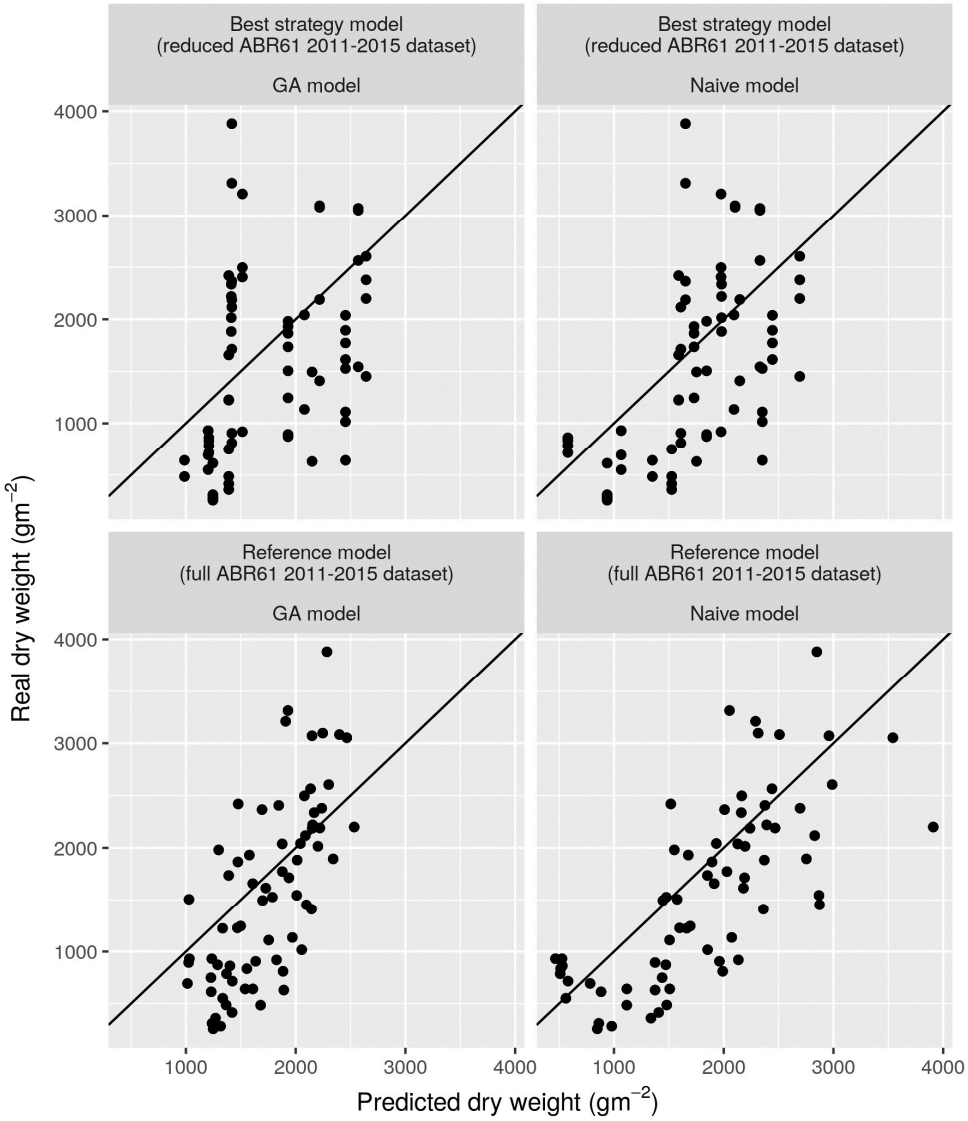

Figure 5.17 Real vs predicted dry weight for the ABR61 dataset, year 2016. Each data point was the dry weight for the plot (not individual plant). The models that made the predictions were part of the scatter search method described in chapter 5. The two compound models (GA and naïve model) and the best data collection strategy were used to train reference and best strategy models – the reference models were trained on the full ABR61 dataset, whereas the best strategy models were trained on the same dataset reduced by the best data collection strategy. This resulted in two best strategy and two reference models, which predicted dry weight in the test ABR61 2016 dataset. The test dataset was not given to the models during training.

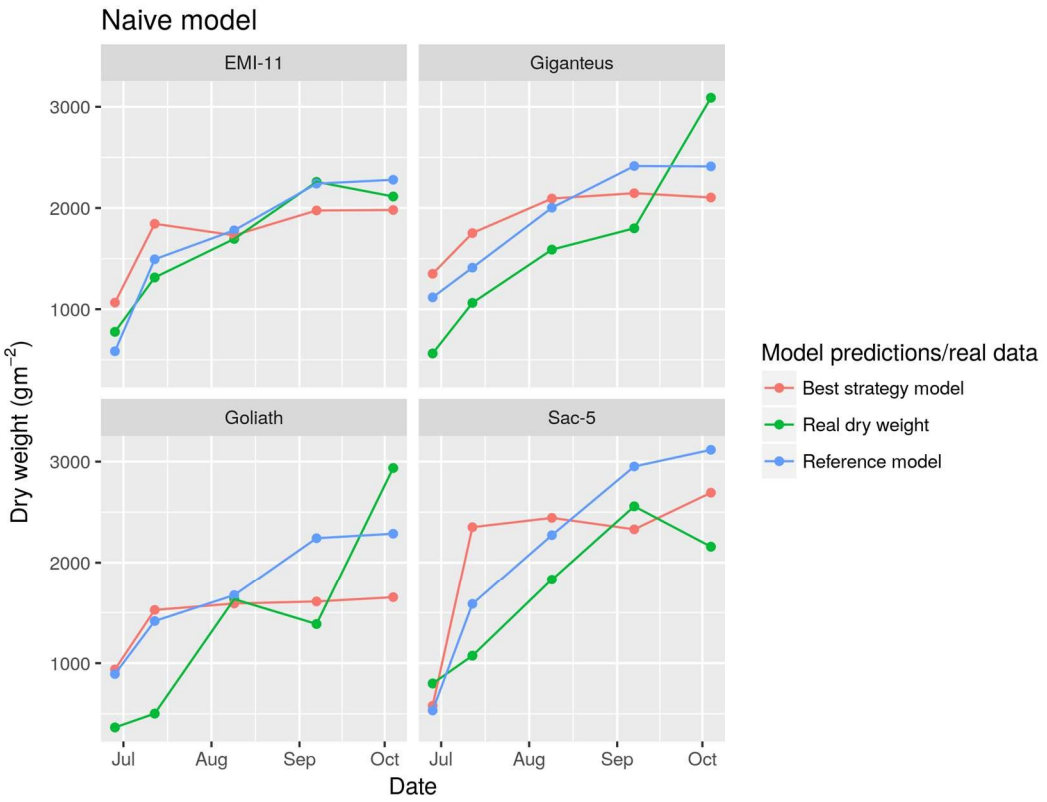

Figure 5.18 Predicted and actual values of dry weight in the ABR61 dataset, year 2016. Each data point was the average dry weight per genotype for that harvest. Real dry weight values from each plot were averaged by genotype. The predictions from the two models (which were plot-level dry weight) were also averaged by genotype. The two models used to make the predictions were the naïve model trained on two versions of the same dataset. The reference model was trained on the ABR61 dataset, years 2011-2015. The best strategy model was trained on the same dataset, reduced by the most optimal data collection strategy. The two models predicted dry weight in year 2016 of the ABR61 trial, plotted here. The real dry weight values are also shown in green.

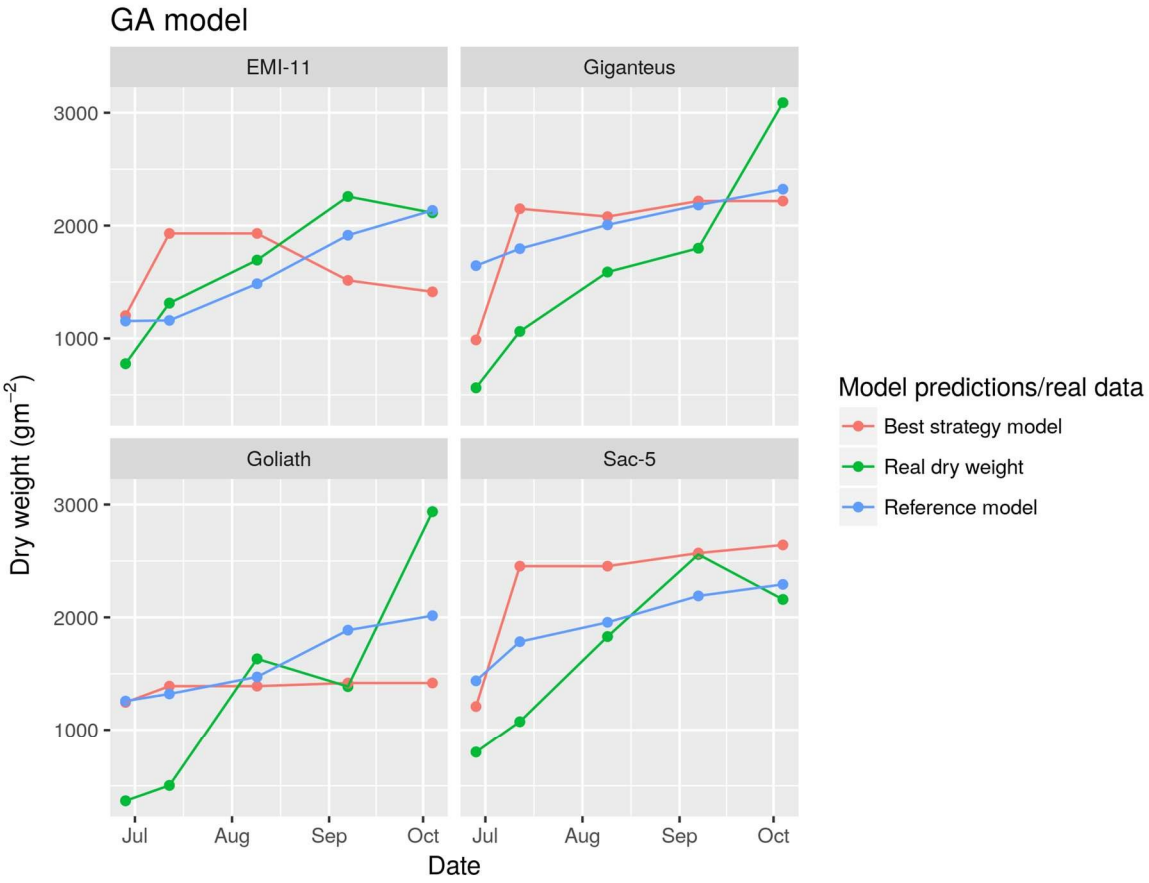

Figure 5.19 Predicted and actual values of dry weight in the ABR61 dataset, year 2016. Each data point was as described in Figure 5.18. The two models used to make the predictions were the GA model trained on two versions of the same dataset. The reference model was trained on the ABR61 dataset, years 2011-2015. The best strategy model was trained on the same dataset, reduced by the most optimal data collection strategy. The two models predicted dry weight in year 2016 of the ABR61 trial, plotted here. The real dry weight values are also shown in green.

Validation

The sample of strategies used for validation was 1500, which was considerably lower than the 10000 strategies used in the previous experiment. This reduction was necessary as the calculation of the $\mathrm{RMSE}_{\mathrm{test}}$ of 10000 strategies with the naïve and GA models would have taken too long. This was because of the considerably longer time it took the naïve and GA models to train, shown on Table 5.5.

Additional differences in the way the scatter search method worked in the two experiments were found during the validation process. As in the previous set of validation results, the Shapiro-Wilk normality test showed that $\mathrm{RMSE}_{\mathrm{train}}$ and $\mathrm{RMSE}_{\mathrm{test}}$ were not normally distributed, and the Wilcoxon signed-rank test found that the two samples were not drawn from the same distribution.

However, the one-sided Wilcoxon signed-rank test showed that the median of $\mathrm{RMSE}_{\mathrm{train}}$ was smaller than the median of $\mathrm{RMSE}_{\mathrm{test}}$, and found the pseudomedian to be -0.209. This meant that according to the statistical test, the cross-validation systematically underestimated the true test error by considerably more compared to the previous experiment. This was also confirmed by the negative median and mean values of the ratio difference distribution listed in Table 5.13. It can be seen from the histogram (Figure 5.20) that the ratio difference distribution was shifted in the negative direction. Compared to the equivalent distribution from the first experiment (Figure 5.12), this one was more spread, which was also confirmed by its higher standard deviation. The small cluster of positive outliers, seen on the first experiment's ratio difference distribution, was not present here. Also, the negative outliers in this experiment's distribution were positioned further away from the mean, almost reaching -1.0.

Median	Mean	SD	N
-0.21	-0.219	0.203	1500

Table 5.13 Statistics for the ratio difference distribution ($RMSE_{train} - RMSE_{test}$). Calculated using the "median", "mean" and "sd" functions in R (R Core Team, 2017).

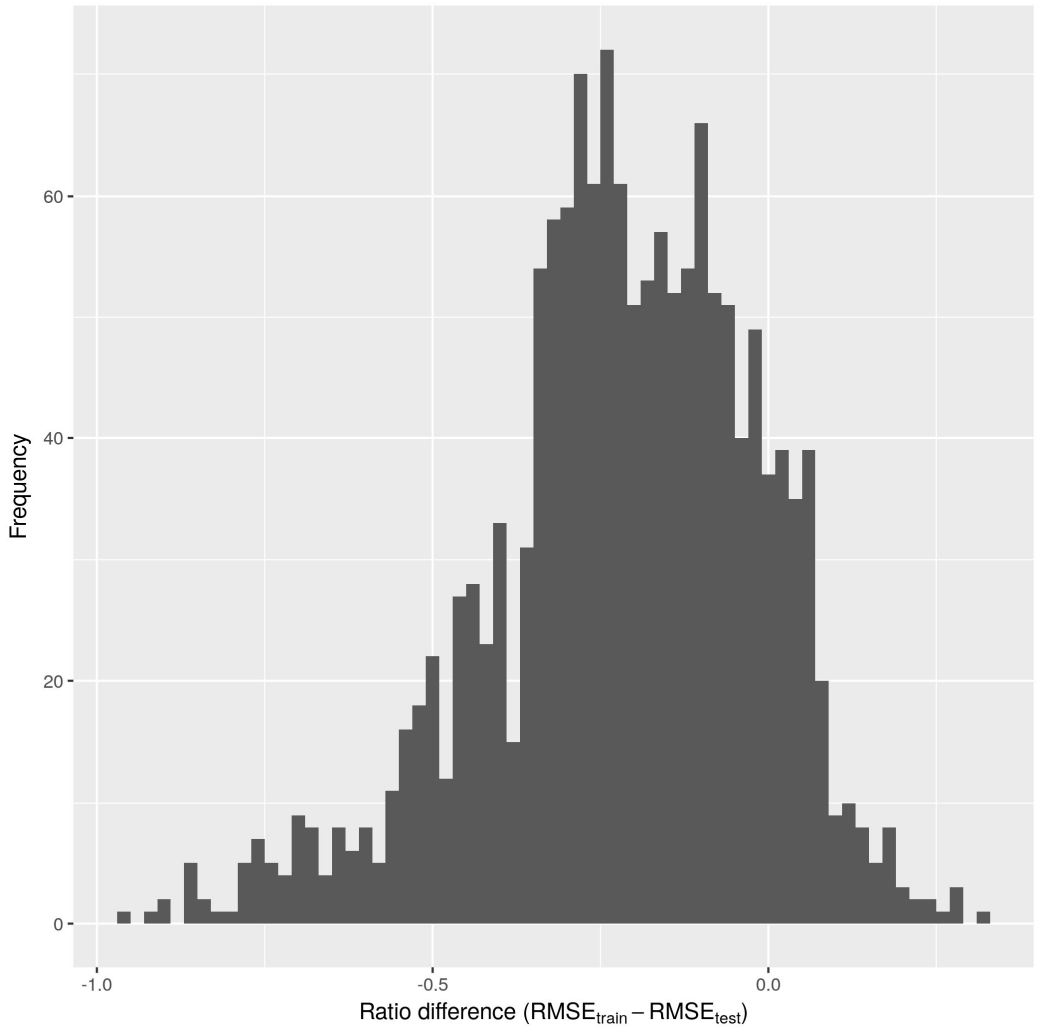

Figure 5.20 Histogram of the differences between RMSE ratios for each of the 1500 data collection strategy samples from the database. The shift of the distribution towards a mean of below 0 meant that the cross-validation RMSE was underestimating the true test RMSE.

The precise cause for the discrepancy between $\mathrm{RMSE}_{\mathrm{train}}$ and $\mathrm{RMSE}_{\mathrm{test}}$ was unclear but it suggested that the naïve and GA models may have been overfitting the training data. A suspected difference which may have caused overfitting in the compound models was their different structure compared to the simple machine learning models. The use of submodels by the compound models may have made these models more likely to overfit, because of their increased complexity, compared to the simple machine learning models (Hawkins, 2004). However, such a conclusion could not be made without further investigation.

The raw RMSE values of the 1500 data strategy samples can be seen in the appendix, section 8.3. Figure 8.3 shows the train RMSE values, obtained from cross-validating each model on the ARB61 2011-2015 dataset. Figure 8.4 shows the test RMSE values, which were calculated from dry weight predictions that these models made on the unseen ABR61 test dataset, year 2016, as well as real dry weight data.

5.4 Discussion

This chapter presented the work related to developing the scatter search-based method for finding the optimal dataset for training accurate models of *Miscanthus* while at the same time minimising the data collection cost. The method was tested against two sets of models developed in the previous chapter of this thesis: simple machine learning models and compound models.

The scatter search method identified the month of May to have a negative effect on model accuracy in both experiments, and it was argued that this was likely due to it being too early in the growing season, when the plants had not grown sufficiently to display enough variability in their phenotype. In both cases the month of July was shown to contain a lot of useful information, whereas for later months (August, September, October) that was less so, although the use of these months by some of the best strategies in both experiments demonstrated that parameter score of 0.5 should not be used as a hard border to decide whether to keep or discard a month.

The same principle was valid for variables in the dataset. LAI was shown to have an adverse effect on accuracy, which combined with its higher cost for collection made it ill-suited for an optimal data collection strategy for training machine learning models. In the two experiments described in this chapter the LAI parameter had a high value score, meaning that it increased

error or increased the cost or both. The best strategies found by the scatter search methods excluded LAI from their list of parameters.

In APBPM, LAI was simulated in order to model the plant light interception. LAI could be replaced by modelling transmission directly in process models, as absolute LAI values are not important in this context, and transmission contains the minimum necessary information for modelling light interception. This was supported by the lower transmission score in both experiments. The results presented in this chapter suggest that collecting LAI in future experiments would take valuable resource but would not provide as much information as transmission.

The decision whether to keep or remove other variables was a lot harder, as their scores were not far enough from 0.5. It was suggested that the influence of these variables on model accuracy was dependent on the presence or absence of other variables. It was impossible to estimate the interaction between the variables using the scatter search algorithm, but it presents an important question for future research in this area.

The limitations of the ABR61 dataset are likely to have influenced the result – particularly for the importance of meteorological variables. As ABR61 was limited to a single location, the conditions experienced by all genotypes were the same. However, if this method was applied to a multi-location trial, it is quite possible that the score value for meteorological variables would be significantly lower than 0.5.

There are other factors that need to be considered when interpreting the results from the two experiments. The two sets of models used different training data, with compound models being able to accept a lot more of the available data points. The models from the two categories had different structures and approaches to modelling, and differed in their ability to produce accurate predictions, as shown in the previous chapter.

The cross-validation procedure had a systematic error when estimating the RMSE of the compound models, which meant that the scatter search procedure, as applied in the second experiment, produced optimistic results for the expected model accuracy for each strategy. This underlined the importance of testing the scatter search method against an unseen test dataset, when applying it to a new model category. The ability of the procedure to find the

optimal dataset was demonstrated for the compound and simple machine learning models, but for a different type of model, its effectiveness needs to be experimentally established.

The method described in this chapter can be a useful tool for selecting optimal variables and measurement times for building precise crop models. Because it takes into account time and not just variables, it is more than just a method for feature selection. While it was applied to models of *Miscanthus* in this chapter, potentially the method could be applicable to datasets and models of other crops. In an example modelling scenario, after gathering an extensive multiyear dataset and developing the crop model, this approach could be applied to the dataset and model to find the optimal dataset for parameterising or training that model over new data. If the scatter search results are verified, then using the optimal strategies and the scores table to plan future data collection should reduce the expenses associated with it, while retaining the modelling accuracy.

Chapter 6: Discussion and future research

6.1 Automatic parameterisation of *Miscanthus* process models

The BSBEC process model (BPM), a *Miscanthus* process model (Davey *et al.*, 2015), was improved in this thesis with an automated parameterisation procedure, which produced the automatically parameterised BPM (APBPM) model. The automated procedure, which was implemented as a software tool written in Python (Python Software Foundation, 2013) and R (R Core Team, 2017), was meant to replace the manual parameterisation of BPM, which was a time-consuming and subjective process.

A comparison between the manually and automatically parameterised models (BPM and APBPM) revealed that while the automated procedure did not make an exact reproduction of BPM when used over the same training data, it actually produced a more accurate model. The values for the light interception k were very close across the models, whereas the radiation use efficiency (RUE) values differed. The two models were shown to be close in their predictions of photosynthetically active radiation (PAR) absorbed by the plant, and dry weight, while their leaf area index (LAI) and light interception values deviated.

This deviation in predictions between the models was caused by the different method employed by APBPM for modelling each genotype's leaf expansion rate (LER), which improved the accuracy of LAI modelling. In fact, all modelled variables: LAI, light interception, PAR and dry weight were simulated more precisely by APBPM, probably due to the use of the sigmoid curve when modelling LER. Changes to the original parameterisation procedure were necessary, as it contained some subjective decisions that could not be automated. They were discarded and replaced with more objective approaches, which overall made APBPM more accurate. Also, one of the aims of the research that produced BPM, was to see whether the parameter values were statistically different across genotypes (Davey *et al.*, 2015). As this was not necessary for the work described in this thesis, APBPM was developed without such statistical comparison.

Another important improvement was that the automated procedure produced a model considerably faster, as it did not require any human intervention. This meant that the model could be parameterised over any compatible dataset, which allows for the expansion of APBPM beyond the four genotypes from the ABR61 dataset. The list of genotypes currently modelled by *Miscanthus* process models is incredibly short. Most models are parameterised only for *Miscanthus x giganteus*: MISCANMOD (Clifton-Brown, Stampfl and Jones, 2004), a model created using the systems software STELLA (Pallipparambil, Raghu and Wiedenmann, 2015), STICS (Strullu *et al.*, 2015) and the model build using SWAT (Ng *et al.*, 2010) are just some of the models limited solely to *M. x giganteus*. MISCANFOR (Hastings *et al.*, 2009), based on MISCANMOD, was parameterised for *M. x giganteus* and partially parameterised for two other *M. sacchariflorus and M. sinensis* genotypes. An earlier version of MISCANMOD (Clifton-Brown *et al.*, 2000) was expanded to model four genotypes: *M. x giganteus* (Gig-2), *M. sacchariflorus* (Sac-5), and two *M. sinensis* hybrids: Sin-H6 and Sin-H9. Finally, BPM was parameterised against four genotypes: *M. x giganteus* (Gig-311), *M. sacchariflorus* (Sac-5), and *M. sinensis* (EMI-11 and Goliath) (Davey *et al.*, 2015).

Crop models are an important tool in plant breeding, particularly in ideotype breeding, which is based upon developing an idea of a hypothetical plant (ideotype) that is the best fit for a target environment (Donald, 1968). Models can describe the ideotype through a set of parameter values and can simulate its likely performance. Beside quantitative traits, model parameters can be used as a target in QTL analysis (Yin *et al.*, 2003). Models also have a high potential to bridge the gap between changes in genetic makeup and phenotypic consequences, especially for complex traits, which are otherwise difficult to select for (Hammer *et al.*, 2006). But to be effective in their application to breeding, *Miscanthus* crop models need to capture the wide variety of genotypic variance in *Miscanthus* (Yan *et al.*, 2012). Therefore, developing models that could quickly be parameterised for multiple genotypes would be an important future step in crop modelling.

The automated parameterisation procedure provides this ability for APBPM, which creates new potential applications of that model. It makes it possible to calculate

heritability of each parameter of the model. Heritability is a statistic, defined as the proportion of phenotypic variability due to genetic effects. It is important in plant breeding, where it is mainly used for estimating the effect of artificial selection for a given trait (Holland, Nyquist and Cervantes-Martinez, 2003). Integrating APBPM in QTL analysis or genomic-wide association study (GWAS) would enable these methods to find genetic loci relevant to genotype specific model parameters (e.g. radiation use efficiency, canopy extinction). This would provide a way to produce genotypes with improved parameter values through breeding, by selecting for these parameters directly.

With the increased availability of genetic data, genomic selection is likely to find adoption in *Miscanthus* breeding programmes in the future (Slavov *et al.*, 2014). Genomic selection models could be used to predict parameter values for new genotypes based on their single nucleotide polymorphism (SNP) data. Finding these parameter values would enable the APBPM model to simulate plant growth for these genotypes under different environmental conditions. This could prove crucial for breeding programmes, as it would drastically reduce the time needed to evaluate new hybrids. However, training accurate crop models requires collection of phenotypic data, but collecting it is a slow and costly process. This problem was addressed in the thesis by investigating its importance to another type of models – machine learning models.

6.2 Screening of machine learning methods for modelling *Miscanthus* yield

Depending on their application, machine learning models can offer a few advantages over mathematical models. They are robust, very fast to train, and can provide accurate predictions, because of their focus on predictive rather than explanatory modelling (Breiman, 2001b; Shmueli, 2010). For these reasons, the problem of training accurate crop models to predict *Miscanthus* yield was examined in the context of supervised machine learning models. First, a wide range of machine learning models, referred to as simple machine learning models, were trained for predicting yield from meteorological and phenotypic data. The models were

compared based on their error, which was established using cross-validation. Another set of models, called compound models, were developed which better resembled the functionality provided by mechanistic crop models – they predicted phenotypic variables from meteorological data, and combined predicted phenotypic and actual meteorological data to predict yield. These models were comprised of several submodels, each responsible for predicting a single variable (canopy height, stem count, etc.). First, a model was created, for which model structure was predetermined and the choice of machine learning algorithms for submodels (referred to as submodel choice) was done using cross-validation. Then, another model was made, where both structure and submodel choice were done using genetic algorithms (GA) (Mitchell, 1996). The model structure and submodel choice discovered in this work were also used in chapter 5, for the compound models trained there.

Among the simple machine learning models, random forests (RF) (Breiman, 2001a), k-nearest neighbour (k-NN) (Fix and Hodges, 1989) and gradient boosted models (GBM) (Natekin and Knoll, 2013) were the most accurate. They outperformed APBPM, although that comparison was unfair as APBPM relied solely on meteorological input data for generating its predictions, whereas the simple machine learning models used phenotypic data as well. The two compound models: naïve model and GA model were shown to slightly outperform APBPM in terms of error, although the GA model's predictions were less correlated with the real data.

A major limitation of the simple machine learning models was their reliance on phenotypic data for generating predictions. This meant that they could not be used for simulating hypothetical scenarios under different environmental conditions. However, it was argued that being accurate models, they still were useful for determining optimal training datasets, a problem which was examined in chapter 5. Solving it could potentially alleviate the phenotyping bottleneck in modelling, as resources for data collection would be well allocated. As it was considered that changes in the training data would affect the information content of the dataset, which would in turn influence model performance, the most accurate models were

needed, as they would have been the most sensitive to changes in the information content.

Machine learning models were also successfully utilised as submodels of the compound models. Besides being more accurate than APBPM, the compound models were more flexible, as they could easily incorporate new variables. While the addition of a variable in a crop model may require redesign of the model structure, as the new variable may affect other stages of the model, the addition of a new variable in the compound models is a straightforward task that can be automated through software. With the expansion of availability of phenotypic data (Kumar *et al.*, 2016), crop models will need to be able to adapt. The flexibility and accuracy of machine learning models gives them an advantage in such applications. Also, as more genetic data is collected in *Miscanthus* (Ma *et al.*, 2012), the same method could be employed for incorporating SNP data into the models, as additional input variables. This could potentially further increase the models' prediction accuracies.

There are many examples where machine learning models already play an important part in crop modelling, particularly where remote sensing data is used (Mathur and Foody, 2008; Kuwata and Shibasaki, 2015; Zheng *et al.*, 2015). Some progress has been made in utilising machine learning with problems usually tackled by regular crop models. For example, artificial neural networks (ANN) were used to predict leaf area from intercepted PAR (Dunea and Moise, 2008). A recent study used naïve Bayes classifier to predict tomato yield from high-dimensional input data, which among other data included meteorological variables (Qaddoum, 2014). Machine learning models have also been shown to be suitable tools for agricultural planning, by demonstrating their ability to predict yield in multiple crops (Gonzalez-Sanchez, Frausto-Solis and Ojeda-Bustamante, 2014). Based on the use of machine learning demonstrated so far in literature, and the findings in this thesis, it is expected that machine learning will appear more often in crop modelling contexts, and will likely contribute to the accuracy, flexibility, and usefulness of crop models in the future.

6.3 Finding optimal datasets for *Miscanthus* models using scatter search

Regardless of the modelling approach, one of the most important steps to developing accurate crop models is the collection of an informative dataset. Gathering phenotypic data is a costly and time-consuming process, and a bottleneck to crop modelling (Furbank and Tester, 2011). Data collection was examined in this thesis as an optimisation problem, where both cost and model error resulting from a data collection strategy needed to be minimised. A method based on the scatter search evolutionary algorithm was developed, which explored different strategies and tried to intelligently improve existing ones. It considered modelling error in terms of RMSE from cross-validation over the best three simple machine learning models (random forests, k-NN and GBM) and the two compound models in two separate experiments. The optimal submodel choice and model structure found for the two compound models in chapter 4 was used in chapter 5 as well. The method built a score table for each variable in the training dataset, as well as each month that measurements were taken in.

The results from both experiments showed the month of July to be distinctly important for the useful information content in the dataset. One of the experiments showed the month of May, and the other May and June, to be unfavourable for taking measurements, as these data points increased the model error. It was argued that this was due to this time being too early in the season, when the plants have not grown enough, whereas July was the month when the most vigorous growth was observed. Both experiments demonstrated that including LAI measurements in the training data had an adverse effect as it increased model error and had the highest associated cost among all variables.

For the rest of the variables, the influence on cost and error was not so clear. The scores that the method assigned each of these variables, were close to 0.5, which denoted the border between positive and negative effect. Additionally, some variables had scores above 0.5 in one experiment, which meant they had a negative effect, whereas in the other they were below 0.5, meaning they were positive. This

disagreement led to the conclusion that the difference between the score of each variable and the 0.5 border needed to be considered when interpreting the results, as it quantifies the influence of that variable over the cost and error. Also, the 0.5 score should not be taken as a hard border, above which all variables should be excluded, but rather as a region whether the effect of the variable is not certain.

The inconclusive results found for most variables suggested that the scatter search method for considering variable influence over error may not be able to completely uncover its underlying mechanism. The method only considered the influence over error of each variable independently, whereas there may have been interaction between variables which influenced error. For example, two variables may bring the same information and there will be no need to use them both, or the positive effect of one variable may turn negative in the presence of another. Further research is needed to establish whether such interactions exist, which would likely improve our current understanding of the importance of each variable. Developing a better method for calculating variable scores, which can investigate variable interactions, would be an important first step in that direction. It could be applied over the database of strategies generated by the scatter search method. Such a method could also be incorporated into the scatter search algorithm, replacing its variable score calculating procedure. Variable importance scores may also vary between models, due to differences in the model algorithm and structure. This could be investigated in the future by comparing variable scores for individual models.

The scatter search method developed in this thesis presents an innovative way to planning data collection, minimising phenotyping costs and relieving the phenotyping bottleneck for projects aiming to build accurate crop models. Similar efforts have been made in crop modelling with remote sensing data, where informative data samples were intelligently picked from the training data, to form smaller datasets that in some cases even improved the modelling accuracy of SVM models (Foody and Mathur, 2004; Mathur and Foody, 2008; Zheng *et al.*, 2015).

Phenotypic data is likely to become more highly dimensional in the future due to the recent advancements in phenomics, through the use of high-throughput phenotyping platforms, which have the ability to automatically collect large amounts of

phenotypic data (Furbank and Tester, 2011; Fahlgren, Gehan and Baxter, 2015). In *Miscanthus* breeding and modelling, it is important to evaluate the plants under field conditions, where they compete for food, water and light with their neighbours. Phenomics in controlled environments is therefore undesirable, as the plants are restricted by the pot they grow in. The application of field-based phenomics in other crops (Jiang, Li and Paterson, 2016) suggests that similar approaches might be used with *Miscanthus* in the future. However, there are still significant challenges in these approaches, that require innovations in the integration of multiple sensors and analysis of the resulting large volumes of data (White *et al.*, 2012). Also, this will not eliminate the need for conventional field measurements, as they will be required for sensor calibration. In this probable scenario, the scatter search method and similar approaches have the potential to reduce the cost for obtaining model training data even further, while at the same time improve modelling accuracy.

In summary, this thesis addressed the issue of phenotypic data collection being the bottleneck in crop modelling, by developing a method for investigating the effect of different data collection strategies on modelling accuracy and data collection cost. It identified both good and bad periods of the growing season for collecting data, as well as measured variables with positive and negative effects. However, further improvements to the method are needed to investigate the effect of some variables with more certainty. A possible solution to the problem of applying crop models to many genotypes was presented, by developing a fast and automatic training procedure for the mechanistic crop model BPM. Finally, the problem of robust, flexible and accurate crop model development was approached by employing various machine learning algorithms and approaches to build predictive models.

References

Adati, S. (1958) ‘Studies on the genus *Miscanthus* with special reference to the Japanese species suitable for breeding purposes as fodder crops’, *Bulletin of the Faculty of Agriculture, Mie University*, 17, pp. 1–112.

Ahlgren, P., Jarneving, B. and Rousseau, R. (2003) ‘Requirements for a cocitation similarity measure, with special reference to Pearson’s correlation coefficient’, *Journal of the American Society for Information Science and Technology*, 54(6), pp. 550–560. doi: 10.1002/asi.10242.

Akinbami, J. F. K., Salami, A. T. and Siyanbola, W. O. (2003) ‘An integrated strategy for sustainable forest-energy-environment interactions in Nigeria’, *Journal of Environmental Management*, 69(2), pp. 115–128. doi: 10.1016/S0301-4797(03)00083-5.

Alexander, D. L. J., Tropsha, A. and Winkler, D. A. (2015) ‘Beware of R(2): simple, unambiguous assessment of the prediction accuracy of QSAR and QSPR models’, *Journal of chemical information and modeling*, 55(7), pp. 1316–1322. doi: 10.1021/acs.jcim.5b00206.

Ali, I., Greifeneder, F., Stamenkovic, J., Neumann, M. and Notarnicola, C. (2015) ‘Review of machine learning approaches for biomass and soil moisture retrievals from remote sensing data’, *Remote Sensing*. doi: 10.3390/rs71215841.

Alpaydin, E. (2010) *Introduction to machine learning*. 2nd edn. The MIT Press.

Andrianasolo, F. N., Casadebaig, P., Maza, E., Champolivier, L., Maury, P. and Debaeke, P. (2014) ‘Prediction of sunflower grain oil concentration as a function of variety, crop management and environment using statistical models’, *European Journal of Agronomy*, 54, pp. 84–96. doi: 10.1016/j.eja.2013.12.002.

Angermueller, C., Pärnamaa, T., Parts, L. and Stegle, O. (2016) ‘Deep learning for computational biology’, *Molecular Systems Biology*, 12(7).

Arnold, J. G. and Fohrer, N. (2005) ‘SWAT2000: Current capabilities and research opportunities in applied watershed modelling’, *Hydrological Processes*, 19(3), pp.

563–572. doi: 10.1002/hyp.5611.

Atkinson, C. J. (2009) 'Establishing perennial grass energy crops in the UK: A review of current propagation options for *Miscanthus*', *Biomass and Bioenergy*, 33(5), pp. 752–759. doi: 10.1016/j.biombioe.2009.01.005.

Barbier, E. (2002) 'Geothermal energy technology and current status: an overview', *Renewable and Sustainable Energy Reviews*, 6(1), pp. 3–65. doi: 10.1016/S1364-0321(02)00002-3.

Battaglini, A., Lilliestam, J., Haas, A. and Patt, A. (2009) 'Development of SuperSmart Grids for a more efficient utilisation of electricity from renewable sources', *Journal of Cleaner Production*, 17(10), pp. 911–918. doi: 10.1016/j.jclepro.2009.02.006.

Battude, M., Al Bitar, A., Morin, D., Cros, J., Huc, M., Marais Sicre, C., Le Dantec, V. and Demarez, V. (2016) 'Estimating maize biomass and yield over large areas using high spatial and temporal resolution Sentinel-2 like remote sensing data', *Remote Sensing of Environment*. Elsevier, 184, pp. 668–681. doi: 10.1016/j.rse.2016.07.030.

Beckmann, J. S. and Soller, M. (1986) 'Restriction fragment length polymorphisms and genetic improvement of agricultural species', *Euphytica*, 35(1), pp. 111–124. doi: 10.1007/BF00028548.

Bernardo, R. (2002) *Breeding for quantitative traits in plants*. Woodbury, MI: Stemma Press.

Bhattacharya, S. C., Attalage, R. A., Augustus Leon, M., Amur, G. Q., Salam, P. A. and Thanawat, C. (1999) 'Potential of biomass fuel conservation in selected Asian countries', *Energy Conversion and Management*, 40(11), pp. 1141–1162. doi: 10.1016/S0196-8904(99)00002-3.

Binongo, J. N. G., Taylor, A., Hill, A. N., Schmotzer, B., Halkar, R., Folks, R., Dubovsky, E., Garcia, E. V. and Manatunga, A. K. (2007) 'Use of classification and regression trees in diuresis renography', *Academic Radiology*, 14(3), pp. 306–311. doi: 10.1016/j.acra.2006.12.013.

Bishop, C. M. (2006) *Pattern recognition and machine learning*, *Pattern Recognition*.

doi: 10.1117/1.2819119.

Boersma, N. N. and Heaton, E. A. (2012) 'Effects of temperature, illumination and node position on stem propagation of *Miscanthus × giganteus*', *GCB Bioenergy*, 4(6), pp. 680–687. doi: 10.1111/j.1757-1707.2011.01148.x.

Boote, K. J., Jones, J. W. and Pickering, N. B. (1996) 'Potential uses and limitations of crop models', *Agronomy Journal*. Madison, WI: American Society of Agronomy, 88, pp. 704–716. doi: 10.2134/agronj1996.00021962008800050005x.

Braga, P. L., Oliveira, A. L. I., Ribeiro, G. H. T. and Meira, S. R. L. (2007) 'Bagging predictors for estimation of software project effort', in *2007 International Joint Conference on Neural Networks*, pp. 1595–1600. doi: 10.1109/IJCNN.2007.4371196.

Breiman, L. (1998) 'Bias, variance and arcing classifiers', *Annals of Statistics*, 26(3), pp. 801–849. doi: 10.1214/aos/1024691079.

Breiman, L. (2001a) 'Random forests', *Machine Learning*, 45(1), pp. 5–32. doi: 10.1023/A:1010933404324.

Breiman, L. (2001b) 'Statistical modeling: the two cultures', *Statistical Science*, 16(3), pp. 199–215. doi: 10.2307/2676681.

Breit, K., House, H., Samson, J., Horkan, A., Harding, T., Dexter, M. and Breuer, H. (2007) 'Dia'. Available at: http://dia-installer.de/.

Breton, S.-P. and Moe, G. (2009) 'Status, plans and technologies for offshore wind turbines in Europe and North America', *Renewable Energy*, 34(3), pp. 646–654. doi: 10.1016/j.renene.2008.05.040.

Cai, X., Zhang, X. and Wang, D. (2011) 'Land availability for biofuel production', *Environmental Science & Technology*. American Chemical Society, 45(1), pp. 334–339. doi: 10.1021/es103338e.

Canteri, M. G., Ávila, B. C., Santos, E. L. dos, Sanches, M. K., Kovalechyn, D. and Gimenez, J. P. M. and L. M. (2002) 'Application of data mining in automatic description of yield behavior in agricultural areas', *Pp. 183-189 in Proceedings of the World Congress of Computers in Agriculture and Natural Resources (13-15, March*

2002, Iguacu Falls, Brazil). St. Joseph, Mich.: ASABE. doi: 10.13031/2013.8328.

Caruana, R., Karampatziakis, N. and Yessenalina, A. (2008) 'An empirical evaluation of supervised learning in high dimensions', *Proceedings of the 25th international conference on Machine learning - ICML '08*, pp. 96–103. doi: 10.1145/1390156.1390169.

Caruana, R. and Niculescu-Mizil, A. (2006) 'An empirical comparison of supervised learning algorithms', *Proceedings of the 23rd international conference on Machine learning*, C(1), pp. 161–168. doi: 10.1145/1143844.1143865.

Chan, H.-Y., Riffat, S. B. and Zhu, J. (2010) 'Review of passive solar heating and cooling technologies', *Renewable and Sustainable Energy Reviews*, 14(2), pp. 781–789. doi: 10.1016/j.rser.2009.10.030.

Chang, Y.-P. C., Kim, J. D.-O., Schwander, K., Rao, D. C., Miller, M. B., Weder, A. B., Cooper, R. S., Schork, N. J., Province, M. a, Morrison, A. C., Kardia, S. L. R., Quertermous, T. and Chakravarti, A. (2006) 'The impact of data quality on the identification of complex disease genes: experience from the Family Blood Pressure Program.', *European journal of human genetics : EJHG*, 14(4), pp. 469–477. doi: 10.1038/sj.ejhg.5201582.

Chapelle, O., Vapnik, V., Bousquet, O. and Mukherjee, S. (2002) 'Choosing multiple parameters for Support Vector Machines', *Machine Learning*, 46(1), pp. 131–159. doi: 10.1023/A:1012450327387.

Chapman, A., Pantalone, V. R., Ustun, A., Allen, F. L., Landau-Ellis, D., Trigiano, R. N. and Gresshoff, P. M. (2003) 'Quantitative trait loci for agronomic and seed quality traits in an F2 and F4:6 soybean population', *Euphytica*, 129(3), pp. 387–393. doi: 10.1023/A:1022282726117.

CIBSE TM41 (2006) *Degree-days: theory and application.*, *Charted Institution of Building Services Engineers, London (2006)*.

Clifton-Brown, J., Neilson, B., Lewandowski, I. and Jones, M. B. B. (2000) 'The modelled productivity of *Miscanthus x giganteus* (GREEF et DEU) in Ireland', *Industrial Crops and Products*, 12(2), pp. 97–109. doi: 10.1016/S0926-

6690(00)00042-X.

Clifton-Brown, J., Lewandowski, I., Andersson, B., Basch, G., Christian, D. G., Kjeldsen, J. B., Jørgensen, U., Mortensen, J. V, Riche, A. B., Schwarz, K.-U., Tayebi, K. and Teixeira, F. (2001) 'Performance of 15 *Miscanthus* genotypes at five sites in Europe', *Agronomy Journal*. Madison, WI: American Society of Agronomy, 93, pp. 1013–1019. doi: 10.2134/agronj2001.9351013x.

Clifton-Brown, J., Robson, P., Allison, G., Lister, S., Sanderson, R., Morris, C., Hodgson, E., Farrar, K., Hawkins, S., Jensen, E., Jones, S., Huang, L., Roberts, P., Youell, S., Jones, B., Wright, A., Valentine, J. and Donnison, I. (2008) '*Miscanthus* : Breeding our way to a better future', in *Biomass and Energy Crops III*, pp. 199–206.

Clifton-Brown, J., Hastings, A., Mos, M., McCalmont, J. P., Ashman, C., Flavell, R., *et al.* (2016) 'Progress in upscaling Miscanthus biomass production for the European bio-economy with seed-based hybrids', *GCB Bioenergy*. Wiley/Blackwell (10.1111), 9(1), pp. 6–17. doi: 10.1111/gcbb.12357.

Clifton-Brown, J., Hastings, A., Mos, M., McCalmont, J. P., Ashman, C., Flavell, R., *et al.* (2017) 'Progress in upscaling *Miscanthus* biomass production for the European bio-economy with seed-based hybrids', *GCB Bioenergy*, 9(1), pp. 6–17. doi: 10.1111/gcbb.12357.

Clifton-Brown, J. and Lewandowski, I. (2000) 'Water use efficiency and biomass partitioning of three different *Miscanthus* genotypes with limited and unlimited water supply', *Annals of Botany*, 86(1), pp. 191–200. doi: 10.1006/anbo.2000.1183.

Clifton-Brown, J. and Lewandowski, I. (2002) 'Screening *Miscanthus* genotypes in field trials to optimise biomass yield and quality in southern germany', *European Journal of Agronomy*, 16(2), pp. 97–110. doi: 10.1016/S1161-0301(01)00120-4.

Clifton-Brown, J., Schwarz, K.-U. and Hastings, A. (2015) 'History of the development of *Miscanthus* as a bioenergy crop: from small beginnings To potential realisation', *Biology and Environment-Proceedings of the Royal Irish Academy*. Royal Irish Academy, 115B(1), pp. 45–57. doi: 10.3318/BIOE.2015.05.

Clifton-Brown, J., Stampfl, P. F. and Jones, M. B. (2004) '*Miscanthus* biomass

production for energy in Europe and its potential contribution to decreasing fossil fuel carbon emissions', *Global Change Biology*, 10(2004), pp. 509–518. doi: 10.1111/j.1529-8817.2003.00749.x.

Collard, B. C. Y., Jahufer, M. Z. Z., Brouwer, J. B. and Pang, E. C. K. (2005) 'An introduction to markers, quantitative trait loci (QTL) mapping and marker-assisted selection for crop improvement: The basic concepts', *Euphytica*, 142(1), pp. 169–196. doi: 10.1007/s10681-005-1681-5.

Collard, B. C. Y. and Mackill, D. J. (2008) 'Marker-assisted selection: an approach for precision plant breeding in the twenty-first century', *Philosophical Transactions of the Royal Society B: Biological Sciences*, 363(1491), p. 557 LP-572. Available at: http://rstb.royalsocietypublishing.org/content/363/1491/557.abstract.

Collins, F. S., Brooks, L. D. and Chakravarti, A. (1998) 'A DNA polymorphism discovery resource for research on human genetic variation', *Genome Research*, 8(12), pp. 1229–1231. doi: 10.1101/gr.8.12.1229.

Cortes, C. and Vapnik, V. (1995) 'Support-Vector Networks', *Machine Learning*, 20(3), pp. 273–297. doi: 10.1023/A:1022627411411.

Craufurd, P. Q., Vadez, V., Jagadish, S. V. K., Prasad, P. V. V. and Zaman-Allah, M. (2013) 'Crop science experiments designed to inform crop modeling', *Agricultural and Forest Meteorology*, 170, pp. 8–18. doi: 10.1016/j.agrformet.2011.09.003.

Cuthbert, J. L., Somers, D. J., Brûlé-Babel, A. L., Brown, P. D. and Crow, G. H. (2008) 'Molecular mapping of quantitative trait loci for yield and yield components in spring wheat (*Triticum aestivum* L.)', *Theoretical and Applied Genetics*, 117(4), pp. 595–608. doi: 10.1007/s00122-008-0804-5.

Dahikar, S. and Rode, S. (2014) 'Agricultural crop yield prediction using artificial neural network approach', *International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering*, 2(1), pp. 2321–2004.

Dai, C., Yao, M., Xie, Z., Chen, C. and Liu, J. (2009) 'Parameter optimization for growth model of greenhouse crop using genetic algorithms', *Applied Soft Computing*, 9(1), pp. 13–19. doi: 10.1016/j.asoc.2008.02.002.

Davey, C. L., Jones, L. E., Squance, M., Purdy, S. J., Maddison, A. L., Cunniff, J., Donnison, I. and Clifton-Brown, J. (2015) 'Radiation capture and conversion efficiencies of *Miscanthus sacchariflorus, M. sinensis* and their naturally occurring hybrid *M. x giganteus*', *GCB Bioenergy*. doi: 10.1111/gcbb.12331.

De'ath, G. (2007) 'Boosted trees for ecological modeling and prediction', *Ecology*, 88(1), pp. 243–251. doi: 10.1890/0012-9658(2007)88[243:BTFEMA]2.0.CO;2.

Delucchi, M. A. and Jacobson, M. Z. (2011) 'Providing all global energy with wind, water, and solar power, Part II: Reliability, system and transmission costs, and policies', *Energy Policy*, 39(3), pp. 1170–1190. doi: 10.1016/j.enpol.2010.11.045.

Denoeux, T. (1995) 'A k-nearest neighbor classification rule based on Dempster-Shafer theory', *IEEE Transactions on Systems, Man, and Cybernetics*, 25(5), pp. 804–813. doi: 10.1109/21.376493.

Deuter, M. (2000) 'Breeding approaches to improvement of yield and quality in *Miscanthus* grown in Europe', *EMI Project, Final report*, pp. 28–52.

Dincer, I. (2000) 'Renewable energy and sustainable development: a crucial review', *Renewable and Sustainable Energy Reviews*, pp. 157–175. doi: 10.1016/S1364-0321(99)00011-8.

Donald, C. M. (1968) 'The breeding of crop ideotypes', *Euphytica*, 17(3), pp. 385–403. doi: 10.1007/BF00056241.

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A. and Vapnik, V. (1997) 'Support vector regression machines', *Advances in neural information processing systems*, 9(x), pp. 155–161. doi: 10.1.1.10.4845.

Drummond, S., Joshi, A. and Sudduth, K. A. (1998) 'Application of neural networks: precision farming', *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)*, pp. 211–215 vol.1. doi: 10.1109/IJCNN.1998.682264.

Dudani, S. A. (1976) 'The distance-weighted k-Nearest-Neighbor rule', *IEEE Transactions on Systems, Man and Cybernetics*, SMC-6(4), pp. 325–327. doi:

10.1109/TSMC.1976.5408784.

Dunea, D. and Moise, V. (2008) 'Artificial neural networks as support for leaf area modelling in crop canopies', in *Proceedings of the 12th WSEAS International Conference on Computers*. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS) (ICCOMP'08), pp. 440–445.

Egré, D. and Milewski, J. C. (2002) 'The diversity of hydropower projects', *Energy Policy*, 30(14), pp. 1225–1230. doi: 10.1016/S0301-4215(02)00083-6.

Ehleringer, J. and Pearcy, R. W. (1983) 'Variation in quantum yield for CO2 uptake among C3 and C4 plants', *Plant Physiology*, 73(3), pp. 555–559. doi: 10.1104/pp.73.3.555.

Ellabban, O., Abu-Rub, H. and Blaabjerg, F. (2014) 'Renewable energy resources: Current status, future prospects and their enabling technology', *Renewable and Sustainable Energy Reviews*, 39, pp. 748–764. doi: 10.1016/j.rser.2014.07.113.

Elzhov, T. V, Mullen, K. M., Spiess, A.-N. and Bolker, B. (2016) 'minpack.lm: R Interface to the Levenberg-Marquardt nonlinear least-squares algorithm found in MINPACK, plus support for bounds'.

European Parliament (2009) 'Directive 2009/28/EC of the European Parliament and of the Council of 23 April 2009', *Official Journal of the European Union*, 140(16), pp. 16–62. doi: 10.3000/17252555.L_2009.140.eng.

Ezaki, B., Nagao, E., Yamamoto, Y., Nakashima, S. and Enomoto, T. (2008) 'Wild plants, *Andropogon virginicus* L. and *Miscanthus sinensis* Anders, are tolerant to multiple stresses including aluminum, heavy metals and oxidative stresses', *Plant Cell Reports*, 27(5), pp. 951–961. doi: 10.1007/s00299-007-0503-8.

Fahlgren, N., Gehan, M. A. and Baxter, I. (2015) 'Lights, camera, action: high-throughput plant phenotyping is ready for a close-up', *Current Opinion in Plant Biology*, 24, pp. 93–99. doi: 10.1016/j.pbi.2015.02.006.

Fang, H. (2003) 'Retrieving leaf area index using a genetic algorithm with a canopy radiative transfer model', *Remote Sensing of Environment*, 85(3), pp. 257–270. doi:

10.1016/S0034-4257(03)00005-1.

Farahnakian, F., Pahikkala, T., Liljeberg, P. and Plosila, J. (2013) 'Energy aware consolidation algorithm based on K-nearest neighbor regression for cloud data centers', in *Proceedings - 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, UCC 2013*, pp. 256–259. doi: 10.1109/UCC.2013.51.

Farrell, A. D., Clifton-Brown, J. C., Lewandowski, I. and Jones, M. B. (2006) 'Genotypic variation in cold tolerance influences the yield of *Miscanthus*', *Annals of Applied Biology*. Blackwell Publishing Ltd, 149(3), pp. 337–345. doi: 10.1111/j.1744-7348.2006.00099.x.

Ferreira, S., Moreira, N. A. and Monteiro, E. (2009) 'Bioenergy overview for Portugal', *Biomass and Bioenergy*, 33(11), pp. 1567–1576. doi: http://dx.doi.org/10.1016/j.biombioe.2009.07.020.

Fischer, G., Prieler, S. and van Velthuizen, H. (2005) 'Biomass potentials of miscanthus, willow and poplar: results and policy implications for Eastern Europe, Northern and Central Asia', *Biomass and Bioenergy*, 28(2), pp. 119–132. doi: 10.1016/j.biombioe.2004.08.013.

Fix, E. and Hodges, J. (1989) 'Discriminatory analysis. Nonparametric discrimination: Consistency properties', *Statistical Review/Revue Internationale de Statistique*, 57, pp. 238–247. doi: 10.2307/1403797.

Foody, G. M. and Mathur, A. (2004) 'Toward intelligent training of supervised image classifications: directing training data acquisition for SVM classification', *Remote Sensing of Environment*, 93(1–2), pp. 107–117. doi: 10.1016/j.rse.2004.06.017.

Freund, Y. and Schapire, R. E. (1997) 'A decision-theoretic generalization of on-line learning and an application to boosting', *Journal of Computer and System Sciences*, 55(1), pp. 119–139. doi: 10.1006/jcss.1997.1504.

Frey, G. W. and Linke, D. M. (2002) 'Hydropower as a renewable and sustainable energy resource meeting global energy challenges in a reasonable way', *Energy Policy*, 30(14), pp. 1261–1265. doi: 10.1016/S0301-4215(02)00086-1.

Fridleifsson, I. B. (2003) 'Status of geothermal energy amongst the world's energy sources', *Geothermics*. Pergamon, 32(4–6), pp. 379–388. doi: 10.1016/J.GEOTHERMICS.2003.07.004.

Friedman, J. H. (2001) 'Greedy function approximation: A gradient boosting machine', *Annals of Statistics*, 29(5), pp. 1189–1232. doi: 10.1214/aos/1013203451.

Friedman, J. H. (2002) 'Stochastic gradient boosting', *Computational Statistics & Data Analysis*, 38(4), pp. 367–378. doi: 10.1016/S0167-9473(01)00065-2.

Friesen, P. C., Peixoto, M. M., Busch, F. A., Johnson, D. C. and Sage, R. F. (2014) 'Chilling and frost tolerance in *Miscanthus* and *Saccharum* genotypes bred for cool temperate climates', *Journal of Experimental Botany*, 65(13), pp. 3749–3758.

Frova, C., Krajewski, P., di Fonzo, N., Villa, M. and Sari-Gorla, M. (1999) 'Genetic analysis of drought tolerance in maize by molecular markers I. Yield components', *Theoretical and Applied Genetics*, 99(1), pp. 280–288. doi: 10.1007/s001220051233.

Fu, M. C., Glover, F. W. and April, J. (2005) 'Simulation optimization: a review, new developments, and applications', in *Proceedings of the Winter Simulation Conference, 2005.*, p. 13 pp.-. doi: 10.1109/WSC.2005.1574242.

Furbank, R. T. and Tester, M. (2011) 'Phenomics – technologies to relieve the phenotyping bottleneck', *Trends in Plant Science*, 16(12), pp. 635–644. doi: 10.1016/j.tplants.2011.09.005.

Gansner, E. R. and North, S. C. (2000) 'An open graph visualization system and its applications to software engineering', *Software - practice and experience*, 30(11), pp. 1203–1233.

Glover, F. (1977) 'Heuristics for integer programming using surrogate constraints', *Decision Sciences*. Wiley/Blackwell (10.1111), 8(1), pp. 156–166. doi: 10.1111/j.1540-5915.1977.tb01074.x.

Glover, F. (1994) 'Genetic algorithms and scatter search: unsuspected potentials', *Statistics and Computing*, 4(2), pp. 131–140. doi: 10.1007/BF00175357.

Glover, F. (1998) 'A template for scatter search and path relinking BT - Artificial

Evolution', in Hao, J.-K. et al. (eds). Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–51.

Goldemberg, J. and Teixeira Coelho, S. (2004) *Renewable energy—traditional biomass vs. modern biomass*, *Energy Policy*. doi: 10.1016/S0301-4215(02)00340-3.

Gonzalez-Sanchez, A., Frausto-Solis, J. and Ojeda-Bustamante, W. (2014) 'Predictive ability of machine learning methods for massive crop yield prediction', *Spanish Journal of Agricultural Research*, 12(2), pp. 313–328. doi: 10.5424/sjar/2014122-4439.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep learning*, *MIT Press*.

Gorenstein Dedecca, J. and Hakvoort, R. A. (2016) 'A review of the North Seas offshore grid modeling: Current and future research', *Renewable and Sustainable Energy Reviews*. Pergamon, 60, pp. 129–143. doi: 10.1016/J.RSER.2016.01.112.

Gortázar, F., Duarte, A., Laguna, M. and Martí, R. (2010) 'Black box scatter search for general classes of binary optimization problems', *Computers & Operations Research*, 37(11), pp. 1977–1986. doi: 10.1016/j.cor.2010.01.013.

Grömping, U. (2006) 'Relative importance for linear regression in R: the package relaimpo', *Journal Of Statistical Software*, 17(1), pp. 139–147. doi: 10.1016/j.foreco.2006.08.245.

Gurung, A. and Oh, S. E. (2013) 'Conversion of traditional biomass into modern bioenergy systems: A review in context to improve the energy situation in Nepal', *Renewable Energy*, 50, pp. 206–213. doi: http://dx.doi.org/10.1016/j.renene.2012.06.021.

Ha, K., Cho, S. and Maclachlan, D. (2005) 'Response models based on bagging neural networks', *Journal of Interactive Marketing*, pp. 17–30. doi: 10.1002/dir.20028.

Hammer, G., Cooper, M., Tardieu, F., Welch, S., Walsh, B., van Eeuwijk, F., Chapman, S. and Podlich, D. (2006) 'Models for navigating biological complexity in breeding improved crop plants', *Trends in Plant Science*, 11(12), pp. 587–593. doi: 10.1016/j.tplants.2006.10.006.

Hammons, T. J. (2003) 'Geothermal power generation worldwide', *2003 IEEE Bologna Power Tech Conference Proceedings,* p. 8 pp. Vol.1. doi: 10.1109/PTC.2003.1304115.

Hastie, T., Tibshirani, R. and Friedman, J. (2009) *The elements of statistical learning: data mining, inference and prediction*. doi: 10.1007/978-0-387-84858-7.

Hastings, A., Clifton-Brown, J., Wattenbach, M., Mitchell, C. P. and Smith, P. (2009) 'The development of MISCANFOR, a new *Miscanthus* crop growth model: towards more robust yield predictions under different climatic and soil conditions', *Global Change Biology - Bioenergy*, 1, pp. 154–170. doi: 10.1111/j.1757-1707.2009.01007.x.

Hastings, A., Mos, M., Yesufu, J. A., McCalmont, J., Schwarz, K., Shafei, R., Ashman, C., Nunn, C., Schuele, H., Cosentino, S., Scalici, G., Scordia, D., Wagner, M. and Clifton-Brown, J. (2017) 'Economic and Environmental Assessment of Seed and Rhizome Propagated Miscanthus in the UK', *Frontiers in Plant Science*, p. 1058. Available at: https://www.frontiersin.org/article/10.3389/fpls.2017.01058.

Hawkins, D. M. (2004) 'The problem of overfitting', *Journal of Chemical Information and Computer Sciences*. American Chemical Society, 44(1), pp. 1–12. doi: 10.1021/ci0342472.

Heaton, E., Voigt, T. and Long, S. P. (2004) 'A quantitative review comparing the yields of two candidate C4 perennial biomass crops in relation to nitrogen, temperature and water', *Biomass and Bioenergy*, 27(1), pp. 21–30. doi: https://doi.org/10.1016/j.biombioe.2003.10.005.

Heffner, E. L., Sorrells, M. E. and Jannink, J.-L. (2009) 'Genomic selection for crop improvement', *Crop Science*. Madison, WI: Crop Science Society of America, 49, pp. 1–12. doi: 10.2135/cropsci2008.08.0512.

Helleputte, T. (2017) 'LiblineaR: Linear predictive models based on the LIBLINEAR C/C++ library'.

Hinton, G. E., Osindero, S. and Teh, Y.-W. (2006) 'A fast learning algorithm for deep belief nets', *Neural Computation*, 18(7), pp. 1527–1554. doi: 10.1162/neco.2006.18.7.1527.

Hodgson, E. M., Nowakowski, D. J., Shield, I., Riche, A., Bridgwater, A. V., Clifton-Brown, J. C. and Donnison, I. S. (2011) 'Variation in *Miscanthus* chemical composition and implications for conversion by pyrolysis and thermo-chemical bio-refining for fuels and chemicals', *Bioresource Technology*. Elsevier, 102(3), pp. 3411–3418. doi: 10.1016/j.biortech.2010.10.017.

Hoerl, A. E. and Kennard, R. W. (1970) 'Ridge Regression: Biased estimation for nonorthogonal problems', *Technometrics*, 12(1), pp. 55–67. doi: 10.1080/00401706.1970.10488634.

Holland, J. B., Nyquist, W. E. and Cervantes-Martinez, C. T. (2003) 'Estimating and interpreting heritability for plant breeding', *Plant Breed. Rev.*, pp. 9–112. doi: 10.1002/9780470650202.ch2.

Humphries, S. W. and Long, S. P. (1995) 'Wimovac - A software package for modeling the dynamics of plant leaf and canopy photosynthesis.', *Computer Applications in the Biosciences*, 11(4), pp. 361–371.

Hyndman, R. J. and Koehler, A. B. (2006) 'Another look at measures of forecast accuracy', *International Journal of Forecasting*, 22(4), pp. 679–688. doi: 10.1016/j.ijforecast.2006.03.001.

Ines, A. V. M., Das, N. N., Hansen, J. W. and Njoku, E. G. (2013) 'Assimilation of remotely sensed soil moisture and vegetation with a crop simulation model for maize yield prediction', *Remote Sensing of Environment*, 138, pp. 149–164. doi: 10.1016/j.rse.2013.07.018.

IPCC (2014) *Climate change 2014: Synthesis report. Contribution of working groups I, II and III to the Fifth assessment report of the Intergovernmental Panel on Climate Change*, *[Core Writing Team, R.K. Pachauri and L.A. Meyer (eds.)]. IPCC, Geneva, Switzerland, 151 pp.* doi: 10.1017/CBO9781107415324.004.

Iqbal, Y., Kiesel, A., Wagner, M., Nunn, C., Kalinina, O., Hastings, A. F. S. J., Clifton-Brown, J. C. and Lewandowski, I. (2017) 'Harvest Time Optimization for Combustion Quality of Different Miscanthus Genotypes across Europe ', *Frontiers in Plant Science* , p. 727. Available at: https://www.frontiersin.org/article/10.3389/fpls.2017.00727.

Iqbal, Y. and Lewandowski, I. (2014) 'Inter-annual variation in biomass combustion quality traits over five years in fifteen *Miscanthus* genotypes in south Germany', *Fuel Processing Technology*, 121, pp. 47–55. doi: 10.1016/j.fuproc.2014.01.003.

van Ittersum, M. ., Leffelaar, P. ., van Keulen, H., Kropff, M. ., Bastiaans, L. and Goudriaan, J. (2003) 'On approaches and applications of the Wageningen crop models', *European Journal of Agronomy*, 18(3–4), pp. 201–234. doi: 10.1016/S1161-0301(02)00106-5.

Jacobson, M. Z. (2009) 'Review of solutions to global warming, air pollution, and energy security', *Energy & Environmental Science*, 2(2), pp. 148–173. doi: 10.1039/B809990C.

Jacobson, M. Z. and Delucchi, M. A. (2011) 'Providing all global energy with wind, water, and solar power, Part I: Technologies, energy resources, quantities and areas of infrastructure, and materials', *Energy Policy*, 39(3), pp. 1154–1169. doi: 10.1016/j.enpol.2010.11.040.

Jame, Y. W. and Cutforth, H. W. (1996) 'Crop growth models for decision support systems', *Canadian Journal of Plant Science*. doi: 10.4141/cjps96-003.

Jamuna, K. S., Karpagavalli, E. S., Vijaya, M. S., Revathi, P., Gokilavani, S. and Madhiya, E. (2010) 'Classification of seed cotton yield based on the growth stages of cotton crop using machine learning techniques', *2010 International Conference on Advances in Computer Engineering*, pp. 312–315. doi: 10.1109/ACE.2010.71.

Jannink, J.-L., Lorenz, A. J. and Iwata, H. (2010) 'Genomic selection in plant breeding: from theory to practice.', *Briefings in functional genomics*, 9(2), pp. 166–177. doi: 10.1093/bfgp/elq001.

Jensen, E., Farrar, K., Thomas-Jones, S., Hastings, A., Donnison, I. and Clifton-Brown, J. (2011) 'Characterization of flowering time diversity in *Miscanthus* species', *GCB Bioenergy*, 3(5), pp. 387–400. doi: 10.1111/j.1757-1707.2011.01097.x.

Jeżowski, S. (2008) 'Yield traits of six clones of *Miscanthus* in the first 3 years following planting in Poland', *Industrial Crops and Products*, 27(1), pp. 65–68. doi: 10.1016/j.indcrop.2007.07.013.

Jiang, Y., Li, C. and Paterson, A. H. (2016) 'High throughput phenotyping of cotton plant height using depth images under field conditions', *Computers and Electronics in Agriculture*, 130, pp. 57–68. doi: 10.1016/j.compag.2016.09.017.

Jing, Q., Conijn, S. J. G., Jongschaap, R. E. E. and Bindraban, P. S. (2012) 'Modeling the productivity of energy crops in different agro-ecological environments', *Biomass and Bioenergy*, 46, pp. 618–633. doi: 10.1016/j.biombioe.2012.06.035.

Jones, M. and Walsh, M. (2013) *Miscanthus: for energy and fibre*. Routledge.

Jørgensen, U. (1997) 'Genotypic variation in dry matter accumulation and content of N, K and Cl in *Miscanthus* in Denmark', *Biomass and Bioenergy*, 12(3), pp. 155–169. doi: 10.1016/S0961-9534(97)00002-0.

Kaldellis, J. K. and Zafirakis, D. (2011) 'The wind energy (r)evolution: A short review of a long history', *Renewable Energy*, 36(7), pp. 1887–1901. doi: 10.1016/j.renene.2011.01.002.

Kalinina, O., Nunn, C., Sanderson, R., Hastings, A. F. S., van der Weijde, T., Özgüven, M., Tarakanov, I., Schüle, H., Trindade, L. M., Dolstra, O., Schwarz, K.-U., Iqbal, Y., Kiesel, A., Mos, M., Lewandowski, I. and Clifton-Brown, J. C. (2017) 'Extending Miscanthus Cultivation with Novel Germplasm at Six Contrasting Sites ', *Frontiers in Plant Science* , p. 563. Available at: https://www.frontiersin.org/article/10.3389/fpls.2017.00563.

Kandel, T. P., Hastings, A., Jørgensen, U. and Olesen, J. E. (2016) *Simulation of biomass yield of regular and chilling tolerant Miscanthus cultivars and reed canary grass in different climates of Europe*, *Industrial Crops and Products*. doi: 10.1016/j.indcrop.2016.04.007.

Kaundal, R., Kapoor, A. S. and Raghava, G. P. S. (2006) 'Machine learning techniques in disease forecasting: a case study on rice blast prediction', *BMC Bioinformatics*, 7(1), p. 485. doi: 10.1186/1471-2105-7-485.

Khanal, R. and Lei, C. (2011) 'Solar chimney—A passive strategy for natural ventilation', *Energy and Buildings*, 43(8), pp. 1811–1819. doi: 10.1016/j.enbuild.2011.03.035.

Klein, T., Calanca, P., Holzkämper, A., Lehmann, N., Roesch, A. and Fuhrer, J. (2012) 'Using farm accountancy data to calibrate a crop model for climate impact studies', *Agricultural Systems*, 111, pp. 23–33. doi: 10.1016/j.agsy.2012.05.001.

Koebner, R. M. D. and Summers, R. W. (2003) '21st century wheat breeding: Plot selection or plate detection?', *Trends in Biotechnology*. Elsevier Current Trends, pp. 59–63. doi: 10.1016/S0167-7799(02)00036-7.

Kohavi, R. (1995) 'A study of Cross-Validation and bootstrap for accuracy estimation and model selection', *International Joint Conference on Artificial Intelligence*, 14(12), pp. 1137–1143. doi: 10.1067/mod.2000.109031.

de Koning, H. W., Smith, K. R. and Last, J. M. (1985) 'Biomass fuel combustion and health.', *Bulletin of the World Health Organization*, 63, pp. 11–26.

Korir, N. K., Han, J., Shangguan, L., Wang, C., Kayesh, E., Zhang, Y. and Fang, J. (2013) 'Plant variety and cultivar identification: advances and prospects', *Crit Rev Biotechnol*, 33(2), pp. 111–125. doi: 10.3109/07388551.2012.675314.

Korte, A. and Farlow, A. (2013) 'The advantages and limitations of trait analysis with GWAS: a review', *Plant Methods*, 9(1), p. 29. doi: 10.1186/1746-4811-9-29.

Kuhn, M. (2016) 'Caret: classification and regression training', *https://CRAN.R-project.org/package=caret*.

Kumar, L. S. (1999) 'DNA markers in plant improvement: An overview', *Biotechnology Advances*, 17(2), pp. 143–182. doi: 10.1016/S0734-9750(98)00018-4.

Kumar, S., Raju, D., Sahoo, R. N. and Chinnusamy, V. (2016) 'Phenomics: unlocking the hidden genetic variation for breaking the barriers in yield and stress tolerance', *Indian Journal of Plant Physiology*, 21(4), pp. 409–419. doi: 10.1007/s40502-016-0261-0.

Kurt, I., Ture, M. and Kurum, A. T. (2008) 'Comparing performances of logistic regression classification and regression tree, and neural networks for predicting coronary artery disease', *Expert Syst. Appl.* Tarrytown, NY, USA: Pergamon Press, Inc., 34(1), pp. 366–374. doi: 10.1016/j.eswa.2006.09.004.

Kuwata, K. and Shibasaki, R. (2015) 'Estimating crop yields with deep learning and remotely sensed data', in *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 858–861. doi: 10.1109/IGARSS.2015.7325900.

Kvålseth, T. O. (1985) 'Cautionary note about R 2', *The American Statistician*. Taylor & Francis, 39(4), pp. 279–285. doi: 10.1080/00031305.1985.10479448.

LeCun, Y., Bengio, Y. and Hinton, G. (2015) 'Deep learning', *Nature*. Nature Publishing Group, 521(7553), pp. 436–444.

Lemus, R., Brummer, E. C., Moore, K. J., Molstad, N. E., Burras, C. L. and Barker, M. F. (2002) 'Biomass yield and quality of 20 switchgrass populations in southern Iowa, USA', *Biomass and Bioenergy*, 23(6), pp. 433–442. doi: 10.1016/S0961-9534(02)00073-9.

Leonhard, W. and Grobe, E. M. (2004) 'Sustainable electrical energy supply with wind and pumped storage - a realistic long-term strategy or utopia?', *IEEE Power Engineering Society General Meeting, 2004.* doi: 10.1109/PES.2004.1373049.

Leung, F. H. F., Lam, H. K., Ling, S. H. and Tam, P. K. S. (2003) 'Tuning of the structure and parameters of a neural network using an improved genetic algorithm', *IEEE Transactions on Neural Networks*, 14(1), pp. 79–88. doi: 10.1109/TNN.2002.804317.

Lewandowski, I., Scurlock, J. M. O., Lindvall, E. and Christou, M. (2003) 'The development and current status of perennial rhizomatous grasses as energy crops in the US and Europe', *Biomass and Bioenergy*, 25(4), pp. 335–361. doi: 10.1016/S0961-9534(03)00030-8.

Lewandowski, I. and Kicherer, A. (1997) 'Combustion quality of biomass: practical relevance and experiments to modify the biomass quality of *Miscanthus x giganteus*', *European Journal of Agronomy*, 6(3), pp. 163–177. doi: 10.1016/S1161-0301(96)02044-8.

Lewandowski, I. and Schmidt, U. (2006) 'Nitrogen, energy and land use efficiencies of miscanthus, reed canary grass and triticale as determined by the boundary line approach', *Agriculture, Ecosystems and Environment*, 112(4), pp. 335–346. doi: 10.1016/j.agee.2005.08.003.

Li, X., Liao, H., Fan, C., Hu, H., Li, Y., Li, J., Yi, Z., Cai, X., Peng, L. and Tu, Y. (2016) 'Distinct geographical distribution of the *Miscanthus* accessions with varied biomass enzymatic saccharification', *PLOS ONE*. Public Library of Science, 11(8), p. e0160026.

Liang, G. and Zhang, C. (2010) 'Empirical study of bagging predictors on medical data', *Conferences in Research and Practice in Information Technology Series*, 121, pp. 31–40.

Liaw, A. and Wiener, M. (2002) 'Classification and regression by randomForest', *R News*, 2(3), pp. 18–22.

Linde-Laursen, I. (1993) 'Cytogenetic analysis of *Miscanthus "Giganteus"*, an interspecific hybrid', *Hereditas*, 119(3), pp. 297–300. doi: 10.1111/j.1601-5223.1993.00297.x.

Liu, J., Goering, C. E. and Tian, L. (2001) 'A neural network for setting target corn yields'. St. Joseph, Mich.: ASABE, p. 705. doi: 10.13031/2013.6097.

Ma, X. F., Jensen, E., Alexandrov, N., Troukhan, M., Zhang, L., Thomas-Jones, S., Farrar, K., Clifton-Brown, J., Donnison, I., Swaller, T. and Flavell, R. (2012) 'High resolution genetic mapping by genome sequencing reveals genome duplication and tetraploid genetic structure of the diploid *Miscanthus sinensis*', *PLoS ONE*, 7(3). doi: 10.1371/journal.pone.0033821.

Maity, J. P., Bundschuh, J., Chen, C.-Y. and Bhattacharya, P. (2014) 'Microalgae for third generation biofuel production, mitigation of greenhouse gas emissions and wastewater treatment: Present and future perspectives – A mini review', *Energy*, 78, pp. 104–113. doi: 10.1016/j.energy.2014.04.003.

Malinowska, M., Donnison, I. S. and Robson, P. R. H. (2016) 'Phenomics analysis of drought responses in *Miscanthus* collected from different geographical locations', *GCB Bioenergy*, 9(1). doi: 10.1111/gcbb.12350.

Mamoshina, P., Vieira, A., Putin, E. and Zhavoronkov, A. (2016) 'Applications of deep learning in biomedicine', *Molecular Pharmaceutics*. American Chemical Society, 13(5), pp. 1445–1454. doi: 10.1021/acs.molpharmaceut.5b00982.

Mansfield, E. R. and Helms, B. P. (1982) 'Detecting multicollinearity', *The American Statistician*. American Statistical Association, Taylor & Francis, Ltd., 36(3), pp. 158–160. doi: 10.2307/2683167.

Martí, R. (2006) 'Scatter Search - Wellsprings and Challenges', *European Journal of Operational Research*, 169, pp. 351–358.

Martí, R., Laguna, M. and Glover, F. (2006) 'Principles of scatter search', *European Journal of Operational Research*, 169(2), pp. 359–372. doi: https://doi.org/10.1016/j.ejor.2004.08.004.

Mathur, A. and Foody, G. M. (2008) 'Crop classification by support vector machine with intelligently selected training data for an operational application', *International Journal of Remote Sensing*. Taylor & Francis, 29(8), pp. 2227–2240. doi: 10.1080/01431160701395203.

McKee, T. B., Doesken, N. J. and Kleist, J. (1995) 'Drought monitoring with multiple time scales', in *Proceedings of the 9th Conference on Applied Climatology*. American Meteorological Society Dallas, Boston, MA, pp. 233–236.

McVicker, I. F. G. (1946) 'The calculation and use of degree-days', *J Inst Heat Vent Eng*, 14, pp. 252–299.

Miao, Y., Mulla, D. J. and Robert, P. C. (2006) 'Identifying important factors influencing corn yield and grain quality variability using artificial neural networks', *Precision Agriculture*, 7(2), pp. 117–135. doi: 10.1007/s11119-006-9004-y.

Miguez, F. E., Zhu, X., Humphries, S., Bollero, G. A. and Long, S. P. (2009) 'A semimechanistic model predicting the growth and production of the bioenergy crop *Miscanthus × giganteus*: description, parameterization and validation', *GCB Bioenergy*. Blackwell Publishing Ltd, 1(4), pp. 282–296. doi: 10.1111/j.1757-1707.2009.01019.x.

Minasny, B. and McBratney, A. B. (2008) 'Regression rules as a tool for predicting soil properties from infrared reflectance spectroscopy', *Chemometrics and Intelligent Laboratory Systems*, 94(1), pp. 72–79. doi: 10.1016/j.chemolab.2008.06.003.

Mitchell, M. (1996) 'An introduction to genetic algorithms', *Computers & Mathematics with Applications*, 32(6), p. 133. doi: 10.1016/S0898-1221(96)90227-8.

Mitchell, R. B., Schmer, M. R., Anderson, W. F., Jin, V., Balkcom, K. S., Kiniry, J., Coffin, A. and White, P. (2016) 'Dedicated energy crops and crop residues for bioenergy feedstocks in the central and eastern USA', *BioEnergy Research*, 9(2), pp. 384–398. doi: 10.1007/s12155-016-9734-2.

Mitchell, T. M. (1997) *Machine learning*, *Annual Review Of Computer Science*. doi: 10.1145/242224.242229.

Mohamad, I. Bin and Usman, D. (2013) 'Standardization and its effects on K-means clustering algorithm', *Research Journal of Applied Sciences, Engineering and Technology*, 6(17), pp. 3299–3303.

Mohan, M., Nair, S., Bhagwat, A., Krishna, T. G., Yano, M., Bhatia, C. R. and Sasaki, T. (1997) 'Genome mapping, molecular markers and marker-assisted selection in crop plants', *Molecular Breeding*, 3(2), pp. 87–103. doi: 10.1023/A:1009651919792.

Monsi, M. and Saeki, T. (1953) 'Uber den Lichtfaktor in den Pflanzengesellschaften und seine Bedeutung fur die Stoffproduktion', *Jpn J Bot*, 14, pp. 22–52.

Monteith, J. L. (1996) 'The quest for balance in crop modeling', *Agronomy Journal*. doi: 10.2134/agronj1996.00021962008800050003x.

Monteith, J. L. and Moss, C. J. (1977) 'Climate and the efficiency of crop production in Britain [and discussion]', *Philosophical Transactions of the Royal Society B: Biological Sciences*, 281(980), pp. 277–294. doi: 10.1098/rstb.1977.0140.

Mountrakis, G., Im, J. and Ogole, C. (2011) 'Support vector machines in remote sensing: A review', *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(3), pp. 247–259. doi: 10.1016/j.isprsjprs.2010.11.001.

Muggeo, V. M. R. (2008) 'segmented: An R package to fit regression models with broken-line relationships', *R News*, 8(May), pp. 20–25. doi: 10.1159/000323281.

Naidu, S. L., Moose, S. P., AL-Shoaibi, A. K., Raines, C. A. and Long, S. P. (2003) 'Cold tolerance of C4 photosynthesis in *Miscanthus x giganteus*: adaptation in amounts and

sequence of C4 photosynthetic enzymes.', *Plant physiology*, 132, pp. 1688–1697. doi: 10.1104/pp.103.021790.

Naik, S. N., Goud, V. V., Rout, P. K. and Dalai, A. K. (2010) 'Production of first and second generation biofuels: A comprehensive review', *Renewable and Sustainable Energy Reviews*, 14(2), pp. 578–597. doi: 10.1016/j.rser.2009.10.003.

Natekin, A. and Knoll, A. (2013) 'Gradient boosting machines, a tutorial', *Frontiers in Neurorobotics*, 7(DEC). doi: 10.3389/fnbot.2013.00021.

Ng, T. L., Eheart, J. W., Cai, X. and Miguez, F. (2010) 'Modeling *Miscanthus* in the Soil and Water Assessment Tool (SWAT) to simulate its water quality effects as a bioenergy crop', *Environmental Science and Technology*, 44(18), pp. 7138–7144. doi: 10.1021/es9039677.

Oh, Y.-H., Chung, T.-K., Kim, M.-K. and Jung, H.-K. (1999) 'Optimal design of electric machine using genetic algorithms coupled with direct method', *IEEE Transactions on Magnetics*, 35(3), pp. 1742–1745. doi: 10.1109/20.767366.

Oteng-Darko, P., Yeboah, S., Addy, S. N. T., Amponsah, S. and Danquah, E. O. (2013) 'Crop modeling: a tool for agricultural research – a review', *E3 Journal of Agricultural Research and Development*, 2(1), pp. 1–6.

Pacala, S. and Socolow, R. (2004) 'Stabilization wedges: solving the climate problem for the next 50 years with current technologies.', *Science (New York, N.Y.)*, 305(5686), pp. 968–72. doi: 10.1126/science.1100103.

Pallipparambil, G. R., Raghu, S. and Wiedenmann, R. N. (2015) 'Modeling the biomass production of the biofuel crop *Miscanthus x giganteus*, to understand and communicate benefits and risks in cultivation', *Energy for Sustainable Development*, 27, pp. 63–72. doi: 10.1016/j.esd.2015.04.005.

Di Paola, A., Valentini, R. and Santini, M. (2016) 'An overview of available crop growth and yield models for studies and assessments in agriculture', *Journal of the Science of Food and Agriculture*, pp. 709–714. doi: 10.1002/jsfa.7359.

Park, S., Im, J., Jang, E. and Rhee, J. (2016) 'Drought assessment and monitoring

through blending of multi-sensor indices using machine learning approaches for different climate regions', *Agricultural and Forest Meteorology*, 216, pp. 157–169. doi: 10.1016/j.agrformet.2015.10.011.

Peixoto, M. de M., Friesen, P. C. and Sage, R. F. (2015) 'Winter cold-tolerance thresholds in field-grown *Miscanthus* hybrid rhizomes', *Journal of Experimental Botany*, 66(14), pp. 4415–4425.

Pereyra-Irujo, G. A. and Aguirrezábal, L. A. N. (2007) 'Sunflower yield and oil quality interactions and variability: Analysis through a simple simulation model', *Agricultural and Forest Meteorology*, 143(3–4), pp. 252–265. doi: 10.1016/j.agrformet.2007.01.001.

Pogson, M. (2011) 'Modelling *Miscanthus* yields with low resolution input data', *Ecological Modelling*, 222, pp. 3849–3853. doi: 10.1016/j.ecolmodel.2011.10.008.

Poorter, H., Anten, N. P. R. and Marcelis, L. F. M. (2013) 'Physiological mechanisms in plant growth models: Do we need a supra-cellular systems biology approach?', *Plant, Cell and Environment*, 36(9), pp. 1673–1690. doi: 10.1111/pce.12123.

Price, L., Bullard, M., Lyons, H., Anthony, S. and Nixon, P. (2004) 'Identifying the yield potential of *Miscanthus x giganteus*: An assessment of the spatial and temporal variability of *M. x giganteus* biomass productivity across England and Wales', *Biomass and Bioenergy*, 26(1), pp. 3–13. doi: 10.1016/S0961-9534(03)00062-X.

Python Software Foundation (2013) 'Python language reference, version 2.7', *Python Software Foundation*. doi: https://www.python.org/.

Qaddoum, K. (2014) 'Modified naïve bayes based prediction modeling for crop yield prediction', *International Journal of Biological, Biomolecular, Agricultural, Food and Biotechnological Engineering*. World Academy of Science, Engineering and Technology, pp. 36–39. Available at: http://waset.org/Publications?p=85.

Le Quéré, C., Andrew, R. M., Canadell, J. G., Sitch, S., Korsbakken, J. I., Zaehle, S., *et al.* (2016) 'Global carbon budget 2016', *Earth System Science Data*, 8(2), pp. 605–649. doi: 10.5194/essd-8-605-2016.

Quinlan, J. R. (1986) 'Induction of decision trees', *Machine Learning*, 1(1), pp. 81–106. doi: 10.1023/A:1022643204877.

Quinlan, J. R. (1992) 'Learning with continuous classes', *Machine Learning*, 92, pp. 343–348. doi: 10.1.1.34.885.

Quinlan, J. R. (1993) *C4.5: Programs for machine learning*, *Morgan Kaufmann San Mateo Californ*ia. doi: 10.1016/S0019-9958(62)90649-6.

R Core Team (2017) 'R: A language and environment for statistical computing'. Vienna, Austria. Available at: https://www.r-project.org/.

Raes, D., Steduto, P., Hsiao, T. C. and Fereres, E. (2009) 'Aquacrop-The FAO crop model to simulate yield response to water: II. main algorithms and software description', *Agronomy Journal*. Madison, WI: American Society of Agronomy, 101(3), pp. 438–447. doi: 10.2134/agronj2008.0140s.

Reijnders, L. (2006) 'Conditions for the sustainability of biomass based fuel use', *Energy Policy*, 34, pp. 863–876. doi: 10.1016/j.enpol.2004.09.001.

REN21 (2016) *Renewables 2016 Global Status Report*, *(Paris: REN21 Secretariat)*.

Ribaut, J.-M., Fracheboud, Y., Monneveux, P., Banziger, M., Vargas, M. and Jiang, C. (2007) 'Quantitative trait loci for yield and correlated traits under high and low soil nitrogen conditions in tropical maize', *Molecular Breeding*, 20(1), pp. 15–29. doi: 10.1007/s11032-006-9041-2.

Ribaut, J. M. and Ragot, M. (2007) 'Marker-assisted selection to improve drought adaptation in maize: The backcross approach, perspectives, limitations, and alternatives', in *Journal of Experimental Botany*, pp. 351–360. doi: 10.1093/jxb/erl214.

Ridgeway, G. (2015) 'gbm: Generalized boosted regression models'. Available at: https://cran.r-project.org/package=gbm.

Rieseberg, L. H. and Carney, S. E. (1998) 'Plant hybridization', *New Phytologist*. Cambridge University Press, 140(4), pp. 599–624. doi: 10.1046/j.1469-8137.1998.00315.x.

Robson, P., Jensen, E., Hawkins, S., White, S. R., Kenobi, K., Clifton-Brown, J., Donnison, I. and Farrar, K. (2013) 'Accelerating the domestication of a bioenergy crop: Identifying and modelling morphological targets for sustainable yield increase in *Miscanthus*', *Journal of Experimental Botany*, 64(14), pp. 4143–4155. doi: 10.1093/jxb/ert225.

Robson, P., Farrar, K., Gay, A. P., Jensen, E. F., Clifton-Brown, J. C. and Donnison, I. S. (2013) 'Variation in canopy duration in the perennial biofuel crop *Miscanthus* reveals complex associations with yield.', *Journal of experimental botany*, 64(8), pp. 2373–83. doi: 10.1093/jxb/ert104.

Romieu, I., Riojas-Rodríguez, H., Marrón-Mares, A. T., Schilmann, A., Perez-Padilla, R. and Masera, O. (2009) 'Improved biomass stove intervention in rural Mexico', *American Journal of Respiratory and Critical Care Medicine*, 180(7), pp. 649–656. doi: 10.1164/rccm.200810-1556OC.

Rumpf, T., Mahlein, A.-K., Steiner, U., Oerke, E.-C., Dehne, H.-W. and Plümer, L. (2010) 'Early detection and classification of plant diseases with Support Vector Machines based on hyperspectral reflectance', *Computers and Electronics in Agriculture*, 74(1), pp. 91–99. doi: 10.1016/j.compag.2010.06.009.

Sage, R. F. and Kubien, D. S. (2007) 'The temperature response of C3 and C4 photosynthesis', *Plant, Cell & Environment*. Wiley/Blackwell (10.1111), 30(9), pp. 1086–1106. doi: 10.1111/j.1365-3040.2007.01682.x.

Schliep, K. and Hechenbichler, K. (2016) 'kknn: Weighted k-Nearest Neighbors'. Available at: https://cran.r-project.org/package=kknn.

Schmitt, M. R. and Edwards, G. E. (1981) 'Photosynthetic capacity and nitrogen use efficiency of maize, wheat, and rice: a comparison between C3 and C4 photosynthesis', *Journal of Experimental Botany*, 32(3), pp. 459–466.

Schrijver, A. (1986) 'Theory of linear and integer programming', *Wiley-Interscience series in discrete mathematics and optimization*. doi: 10.1016/0378-4754(87)90121-2.

Shapiro, S. S. and Wilk, M. B. (1965) 'An analysis of variance test for normality

(complete samples)', *Biometrika*. [Oxford University Press, Biometrika Trust], 52(3/4), pp. 591–611. doi: 10.2307/2333709.

Shmueli, G. (2010) 'To explain or to predict?', *Statistical Science*, 25, pp. 289–310. doi: 10.1214/10-STS330.

Sims, R. E. H., Hastings, A., Schlamadinger, B., Taylor, G. and Smith, P. (2006) 'Energy crops: current status and future prospects', *Global Change Biology*. doi: 10.1111/j.1365-2486.2006.01163.x.

Sinclair, T. R. and Seligman, N. G. (1996) 'Crop modeling: From infancy to maturity', *Agronomy Journal*. doi: 10.2134/agronj1996.00021962008800050004x.

Singh, A., Olsen, S. I. and Nigam, P. S. (2011) 'A viable technology to generate third-generation biofuel', *Journal of Chemical Technology & Biotechnology*. John Wiley & Sons, Ltd., 86(11), pp. 1349–1353. doi: 10.1002/jctb.2666.

Slavov, G., Robson, P., Jensen, E., Hodgson, E., Farrar, K., Allison, G., Hawkins, S., Thomas-Jones, S., Ma, X.-F., Huang, L., Swaller, T., Flavell, R., Clifton-Brown, J. and Donnison, I. (2013) 'Contrasting geographic patterns of genetic variation for molecular markers vs. phenotypic traits in the energy grass *Miscanthus sinensis*', *GCB Bioenergy*, 5(5), pp. 562–571. doi: 10.1111/gcbb.12025.

Slavov, G. T., Nipper, R., Robson, P., Farrar, K., Allison, G. G., Bosch, M., Clifton-Brown, J. C., Donnison, I. S. and Jensen, E. (2014) 'Genome-wide association studies and prediction of 17 traits related to phenology, biomass and cell wall composition in the energy grass *Miscanthus sinensis*', *New Phytologist*, 201(4), pp. 1227–1239. doi: 10.1111/nph.12621.

Smola, A. J., Sch, B. and Schölkopf, B. (2004) 'A tutorial on Support Vector Regression', *Statistics and Computing*, 14(3), pp. 199–222. doi: 10.1023/B:STCO.0000035301.49549.88.

Sobol', I. . (2001) 'Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates', *Mathematics and Computers in Simulation*, 55(1–3), pp. 271–280. doi: 10.1016/S0378-4754(00)00270-6.

Spahić, E., Balzer, G., Hellmich, B. and Münch, W. (2007) 'Wind energy storages - Possibilities', in *2007 IEEE Lausanne POWERTECH, Proceedings*, pp. 615–620. doi: 10.1109/PCT.2007.4538387.

Spitters, C. (1987) 'An analysis of variation in yield among potato cultivars in terms of light absorption, light utilization and dry matter partitioning', *Agrometeorology of the Potato Crop 214*, pp. 71–84.

Sravan, T., Jaiswal, H. K., Waza, S. A. and Priyanka, K. (2016) 'Analysis of variability and character association in indigenous aromatic rice (*Oryza sativa* L .)', *Electronic Journal of Plant Breeding*, 7(1), pp. 159–164. doi: 10.5958/0975-928X.2016.00023.5.

Srinivas, M. and Patnaik, L. M. (1994) 'Genetic Algorithms: A Survey', *Computer*. doi: 10.1109/2.294849.

Srirangan, K., Akawi, L., Moo-Young, M. and Chou, C. P. (2012) 'Towards sustainable production of clean energy carriers from biomass resources', *Applied Energy*. Elsevier Ltd, 100, pp. 172–186. doi: 10.1016/j.apenergy.2012.05.012.

Stavridou, E., Hastings, A., Webster, R. J. and Robson, P. R. H. (2017) 'The impact of soil salinity on the yield, composition and physiology of the bioenergy grass *Miscanthus × giganteus*', *GCB Bioenergy*, 9(1), pp. 92–104. doi: 10.1111/gcbb.12351.

Steane, A. M. (1996) 'Error correcting codes in quantum theory', *Physical Review Letters*, 77(July), pp. 793–797. doi: 10.1103/PhysRevLett.77.793.

Steduto, P., Hsiao, T. C., Raes, D. and Fereres, E. (2009) 'AquaCrop—The FAO crop model to simulate yield response to water: I. Concepts and underlying principles', *Agronomy Journal*. Madison, WI: American Society of Agronomy, 101(3), pp. 426–437. doi: 10.2134/agronj2008.0139s.

Steinke, F., Wolfrum, P. and Hoffmann, C. (2013) 'Grid vs. storage in a 100% renewable Europe', *Renewable Energy*, 50, pp. 826–832. doi: 10.1016/j.renene.2012.07.044.

Street, W. N. (2005) 'Oblique multicategory decision trees using nonlinear programming', *INFORMS Journal on Computing*, 17(1), pp. 25–31. doi:

10.1287/ijoc.1030.0047.

Stričević, R., Dželetović, Z., Djurović, N. and Cosić, M. (2015) 'Application of the AquaCrop model to simulate the biomass of *Miscanthus x giganteus* under different nutrient supply conditions', *GCB Bioenergy*, 7(6), pp. 1203–1210. doi: 10.1111/gcbb.12206.

Strullu, L., Beaudoin, N., de Cortàzar Atauri, I. G. and Mary, B. (2014) 'Simulation of biomass and nitrogen dynamics in perennial organs and shoots of *Miscanthus Giganteus* using the STICS model', *BioEnergy Research*, 7(4), pp. 1253–1269. doi: 10.1007/s12155-014-9462-4.

Strullu, L., Ferchaud, F., Yates, N., Shield, I., Beaudoin, N., Garcia de Cortazar-Atauri, I., Besnard, A. and Mary, B. (2015) 'Multisite yield gap analysis of *Miscanthus × giganteus* using the STICS model', *BioEnergy Research*. Springer US, 8(4), pp. 1735–1749. doi: 10.1007/s12155-015-9625-y.

Taramino, G. and Tingey, S. (1996) 'Simple sequence repeats for germplasm analysis and mapping in maize', *Genome*. NRC Research Press, 39(2), pp. 277–287. doi: 10.1139/g96-038.

Theunissen, T. J. J. M. (1985) 'Binary programming and test design', *Psychometrika*, 50(4), pp. 411–420. doi: 10.1007/BF02296260.

Tibshirani, R. (1996) 'Regression selection and shrinkage via the Lasso', *Journal of the Royal Statistical Society B*, pp. 267–288. doi: 10.2307/2346178.

Tsoumakas, G., Partalas, I. and Vlahavas, I. (2009) 'An Ensemble Pruning Primer.', in Okun, O. and Valentini, G. (eds) *Applications of Supervised and Unsupervised Ensemble Methods.* Berlin, Heidelberg: Springer, pp. 1–13. doi: 10.1007/978-3-642-03999-7_1.

United Nations/Framework Convention on Climate Change (2015) 'Paris Agreement', *21st Conference of the Parties*, p. 3. doi: FCCC/CP/2015/L.9.

Valentine, J., Clifton-Brown, J., Hastings, A., Robson, P., Allison, G. and Smith, P. (2011) 'Food vs. fuel: the use of land for lignocellulosic "next generation" energy

crops that minimize competition with primary food production', *GCB Bioenergy*. Wiley/Blackwell (10.1111), 4(1), pp. 1–19. doi: 10.1111/j.1757-1707.2011.01111.x.

Venables, W. N. and Ripley, B. D. (2002) *Modern applied statistics with S*. New York: Springer.

Verhurlst, P. (1838) 'Notice sur la loi que la population suit dans son accroissement.', *Correspondance mathématique et physique publiée par A. Quételet (Brussels)*, (10), pp. 113–121.

Ververis, C., Georghiou, K., Christodoulakis, N., Santas, P. and Santas, R. (2004) 'Fiber dimensions, lignin and cellulose content of various plant materials and their suitability for paper production', *Industrial Crops and Products*, 19(3), pp. 245–254. doi: 10.1016/j.indcrop.2003.10.006.

Vos, P., Hogers, R., Bleeker, M., Reijans, M., Lee, T. van de, Hornes, M., Friters, A., Pot, J., Paleman, J., Kuiper, M. and Zabeau, M. (1995) 'AFLP: a new technique for DNA fingerprinting', *Nucleic Acids Research*, 23(21), pp. 4407–4414.

Walsh, B. (2001) 'Quantitative genetics', in *eLS*. John Wiley & Sons, Ltd. doi: 10.1038/npg.els.0001785.

Waongo, M., Laux, P., Traoré, S. B., Sanon, M. and Kunstmann, H. (2013) 'A crop model and fuzzy rule based approach for optimizing maize planting dates in Burkina Faso, West Africa', *Journal of Applied Meteorology and Climatology*. American Meteorological Society, 53(3), pp. 598–613. doi: 10.1175/JAMC-D-13-0116.1.

White, J. W., Andrade-Sanchez, P., Gore, M. A., Bronson, K. F., Coffelt, T. A., Conley, M. M., Feldmann, K. A., French, A. N., Heun, J. T., Hunsaker, D. J., Jenks, M. A., Kimball, B. A., Roth, R. L., Strand, R. J., Thorp, K. R., Wall, G. W. and Wang, G. (2012) 'Field-based phenomics for plant genetics research', *Field Crops Research*. Elsevier, 133, pp. 101–112. doi: 10.1016/j.fcr.2012.04.003.

Whitley, D., Rana, S., Dzubera, J. and Mathias, K. E. (1996) 'Evaluating evolutionary algorithms', *Artificial Intelligence*, 85(1–2), pp. 245–276. doi: 10.1016/0004-3702(95)00124-7.

Wickham, H. (2009) *ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York. Available at: http://ggplot2.org.

Williams, J. G. K., Kubelik, A. R., Livak, K. J., Rafalski, J. A. and Tingey, S. V (1990) 'DNA polymorphisms amplified by arbitrary primers are useful as genetic markers', *Nucleic Acids Research*, 18(22), pp. 6531–6535.

Wind Europe (2017) *Offshore wind in Europe - Key trends and statistics*, *Refocus*. doi: 10.1016/S1471-0846(02)80021-X.

Wong, M. T. F. and Asseng, S. (2006) 'Determining the causes of spatial and temporal variability of wheat yields at sub-field scale using a new method of upscaling a crop model', *Plant and Soil*, 283(1), pp. 203–215. doi: 10.1007/s11104-006-0012-5.

Woolson, R. F. (2007) 'Wilcoxon signed-rank test', in *Wiley Encyclopedia of Clinical Trials*. John Wiley & Sons, Inc. doi: 10.1002/9780471462422.eoct979.

Xu, Y. (2010) *Molecular plant breeding*. Cabi. doi: 10.1079/9781845933920.0000.

Xu, Y. and Crouch, J. H. (2008) 'Marker-assisted selection in plant breeding: From publications to practice', *Crop Science*, 48(2), pp. 391–407. doi: 10.2135/cropsci2007.04.0191.

Xue, S., Kalinina, O. and Lewandowski, I. (2015) 'Present and future options for Miscanthus propagation and establishment', *Renewable and Sustainable Energy Reviews*, 49, pp. 1233–1246. doi: 10.1016/j.rser.2015.04.168.

Yan, J., Chen, W., Luo, F., Ma, H., Meng, A., Li, X., Zhu, M., Li, S., Zhou, H., Zhu, W., Han, B., Ge, S., Li, J. and Sang, T. (2012) 'Variability and adaptability of *Miscanthus* species evaluated for energy crop domestication', *GCB Bioenergy*. Blackwell Publishing Ltd, 4(1), pp. 49–60. doi: 10.1111/j.1757-1707.2011.01108.x.

Yang, X.-S. (1970) *Optimization Algorithms*. doi: 10.1007/978-3-642-20859-1_2.

Yin, X., Stam, P., Kropff, M. J. and Schapendonk, A. H. C. M. (2003) 'Crop modeling, QTL mapping, and their complementary role in plant breeding', *Agronomy Journal*. Madison, WI: American Society of Agronomy, 95, pp. 90–98. doi: 10.2134/agronj2003.9000.

You, A., Lu, X., Jin, H., Ren, X., Liu, K., Yang, G., Yang, H., Zhu, L. and He, G. (2006) 'Identification of quantitative trait loci across recombinant inbred lines and testcross populations for traits of agronomic importance in rice', *Genetics*. Copyright © 2006 by the Genetics Society of America, 172(2), pp. 1287–1300. doi: 10.1534/genetics.105.047209.

Yu, D. and Deng, L. (2011) 'Deep learning and its applications to signal and information processing', *Signal Processing Magazine, IEEE*, 28(1), pp. 145–154. doi: 10.1109/MSP.2010.939038.

Zaitlen, N. and Kraft, P. (2012) 'Heritability in the genome-wide association era', *Human genetics*, 131(10), pp. 1655–1664. doi: 10.1007/s00439-012-1199-6.

Zhang, S. (2012) 'Nearest neighbor selection for iteratively kNN imputation', *Journal of Systems and Software*, 85(11), pp. 2541–2552. doi: 10.1016/j.jss.2012.05.073.

Zheng, B., Myint, S. W., Thenkabail, P. S. and Aggarwal, R. M. (2015) 'A support vector machine to identify irrigated crop types using time-series Landsat NDVI data', *International Journal of Applied Earth Observation and Geoinformation*, 34, pp. 103–112. doi: 10.1016/j.jag.2014.07.002.

Zirkle, C. (1934) 'Some forgotten records of hybridization and sex in plants, 1716-1739.', *Journal of Heredity*. The University of Chicago Press, 23(11), pp. 433–447. doi: 10.1086/346844.

Appendix

8.1 Simple machine learning models training procedure in R

This section contains the R script that was used to train simple machine learning models:

```
#!/usr/bin/env Rscript
suppressMessages(library(randomForest))
suppressMessages(library(nnet))
suppressMessages(library(caret))
suppressMessages(library(mgcv))
suppressMessages(library(kknn))
suppressMessages(library(e1071))
suppressMessages(library(gbm))
suppressMessages(library(neuralnet))
suppressMessages(library(LiblineaR))
suppressMessages(library(plyr))

normalise_frame <- function(data_frame, reference){
    if (unique(data_frame$pseudo_rep_no) == 0){
        data_frame = subset(data_frame, select = -c(pseudo_rep_no))
    }

    for (name in colnames(data_frame)){
        data_frame[[name]] <- as.numeric(data_frame[[name]])
        if (missing(reference)){
            data_frame[[name]] <- normalise(data_frame[[name]])
        } else {
            data_frame[[name]] <- normalise(data_frame[[name]],
                                            reference[[name]])
        }
    }

    return(data_frame)
}

unnormalise_frame <- function(data_frame, reference){
    numerics = c("dry_weight", "Transmission", "canopy_height", "stem_count", "PAR",
                 "degree_days", "rainfall_mm")
    for (name in numerics){
        if (name %in% colnames(data_frame)){
            if (missing(reference)){
                data_frame[[name]] <- unnormalise(data_frame[[name]])
            } else {
                data_frame[[name]] <- unnormalise(data_frame[[name]],
                                                  reference[[name]])
            }
        }
    }

    return(data_frame)
}

normalise <- function(vect, reference){
    if (missing(reference)){
        x_max = max(vect)
        x_min = min(vect)
    } else {
        x_max = attr(reference, "x_max")
        x_min = attr(reference, "x_min")
    }

    vect = (vect - x_min) / (x_max - x_min)
    attr(vect, "x_max") <- x_max
    attr(vect, "x_min") <- x_min
    return(vect)
}

unnormalise <- function(vect, reference){
```

```
    if (missing(reference)){
        reference = vect
    }

    x_max = attr(reference, "x_max")
    x_min = attr(reference, "x_min")
    vect = (x_max - x_min) * vect + x_min
    return(vect)

}

sample_data <- function(train_data){
    size = nrow(train_data)
    output <- train_data[sample(nrow(train_data), size, replace = TRUE), ]
    if ("genotype" %in% colnames(output)){
        levels(output$genotype) <- c("EMI-11", "Giganteus", "Goliath", "Sac-5")
    }
    return(output)
}

prep data <- function(fname, train data){
    data <- read.csv(fname)

    if ('row' %in% colnames(data)) {
        rc_factors <- 1:8
        data$row <- factor(data$row, levels = rc_factors)
        data$col <- factor(data$col, levels = rc factors)
    }

    if ("genotype" %in% colnames(data)){
        data$genotype <- factor(data$genotype)
        levels(data$genotype) <- c("EMI-11", "Giganteus", "Goliath", "Sac-5")
    }

    if ('field_flowering_score' %in% colnames(data)){
        data$field_flowering_score <- as.numeric(data$field_flowering_score)
    }

    if ("doy" %in% colnames(data)){
        data$doy <- as.numeric(data$doy)
    }

    sub <- subset(data, !is.na(dry_weight))
    return(na.omit(sub))
}

bag_me <- function(train_data, test_data, train_model, predict_function, n){
    if (missing(n)) {
        n = 100
    }
    #train n versions of the model (100 by default)
    models <- lapply(1:n, train_model)

    predictions <- sapply(models, predict_function)
    test_data$predicted <- apply(predictions, 1, mean)

    return(test_data)
}

random_forest_scenario <- function(train_data, test_data){
    rf <- randomForest(dry weight ~ ., data = train data)
    test_data$predicted <- predict(rf, test_data, OOB=TRUE)

    return(test_data)
}

random_forest_scenario_caret <- function(train_data, test_data){
    rf <- train(dry_weight ~ ., data = train_data, method='rf')
    test_data$predicted <- predict(rf, test_data)

    return(test_data)
}

lm_scenario_caret <- function(train_data, test_data){
    fit <- train(dry_weight ~ ., data = train_data, method = 'lm')
    if ("genotype" %in% colnames(train_data)){
        fit$xlevels[["genotype"]] <- c("EMI-11", "Giganteus", "Goliath", "Sac-5")
```

```
    }

    if ("year" %in% colnames(train_data)){
        fit$xlevels[["year"]] <- c(2011, 2015, 2016)
    }

    test_data$predicted <- predict(fit, test_data)

    return(test data)
}

lm_bagging_scenario_caret <- function(train_data, test_data){

    train_model <- function(i){
        train sample <- sample data(train data)
        fit <- train(dry_weight ~ ., data = train_sample, method='lm')
        if ("genotype" %in% colnames(train_sample)){
            fit$xlevels[["genotype"]] <- c("EMI-11", "Giganteus", "Goliath", "Sac-5")
        }

        if ("year" %in% colnames(train sample)){
            fit$xlevels[["year"]] <- c(2011, 2015, 2016)
        }

        return(fit)
    }

    predict_function <- function(model){
        return(predict(model, test_data))
    }

    return(bag_me(train_data, test_data, train_model, predict_function))
}

lm_scenario <- function(train_data, test_data){
    fit <- lm(dry_weight ~ ., data = train_data)
    if ("genotype" %in% colnames(train_data)){
        fit$xlevels[["genotype"]] <- c("EMI-11", "Giganteus", "Goliath", "Sac-5")
    }

    if ("year" %in% colnames(train_data)){
        fit$xlevels[["year"]] <- c(2011, 2015, 2016)
    }

    test_data$predicted <- predict(fit, test_data)

    return(test_data)
}


lm_bagging_scenario <- function(train_data, test_data){

    train_model <- function(i){
        train_sample <- sample_data(train_data)
        fit <- lm(dry_weight ~ ., data = train_sample)
        if ("genotype" %in% colnames(train_sample)){
            fit$xlevels[["genotype"]] <- c("EMI-11", "Giganteus", "Goliath", "Sac-5")
        }

        if ("year" %in% colnames(train_sample)){
            fit$xlevels[["year"]] <- c(2011, 2015, 2016)
        }

        return(fit)
    }

    predict_function <- function(model){
        return(predict(model, test_data))
    }

    return(bag_me(train_data, test_data, train_model, predict_function))
}

ann_scenario_neuralnet <- function(train_data, test_data){
    trd <- normalise_frame(train_data)
    ted <- normalise_frame(test_data, trd)
    capture.output(fit <- train(dry_weight ~ ., data = trd, method='neuralnet'))
```

```
    ted$predicted <- predict(fit, ted)
    test_data$predicted <- unnormalise(ted$predicted, ted$dry_weight)
    return(test data)
}

ann_bagging_scenario_neuralnet <- function(train_data, test_data){
    trd <- normalise_frame(train_data)
    ted <- normalise_frame(test data, trd)

    train_model <- function(i){
        train_sample <- sample_data(trd)
        capture.output(fit <- train(dry_weight ~ ., data = train_sample,
                                    method='neuralnet'))
        return(fit)
    }

    predict_function <- function(model){
        return(predict(model, ted))
    }

    ted <- bag_me(trd, ted, train_model, predict_function, n = 10)

    test_data$predicted <- unnormalise(ted$predicted, ted$dry_weight)
    return(test_data)
}

ann_scenario <- function(train_data, test_data){
    train_data$dry_weight <- normalise(train_data$dry_weight)
    test_data$dry_weight <- normalise(test_data$dry_weight, train_data$dry_weight)

    capture.output(fit <- train(dry_weight ~ ., data = train_data, method = 'nnet'))
    test_data$predicted <- predict(fit, test_data)

    test_data$dry_weight <- unnormalise(test_data$dry_weight)
    test_data$predicted <- unnormalise(test_data$predicted, test_data$dry_weight)

    return(test data)
}

ann_bagging_scenario <- function(train_data, test_data){
    train_data$dry_weight <- normalise(train_data$dry_weight)
    test_data$dry_weight <- normalise(test_data$dry_weight, train_data$dry_weight)

    train_model <- function(i){
        train_sample <- sample_data(train_data)
        capture.output(fit <- train(dry_weight ~ ., data = train_sample, method =
'nnet'))
        return(fit)
    }

    predict_function <- function(model){
        return(predict(model, test_data))
    }

    test_data <- bag_me(train_data, test_data, train_model, predict_function, n = 10)
    test_data$dry_weight <- unnormalise(test_data$dry_weight)
    test_data$predicted <- unnormalise(test_data$predicted, test_data$dry_weight)

    return(test_data)

}

glm_scenario <- function(train_data, test_data){
    fit <- glm(dry_weight ~ .,, data = train_data)
    if ("year" %in% colnames(train_data)){
        fit$xlevels[["year"]] <- c(2011, 2015, 2016)
    }
    test_data$predicted <- predict(fit, test_data)

    return(test_data)
}

knn_scenario <- function(train_data, test_data){
    fit <- kknn(dry_weight ~ .,, train = train_data, test = test_data)
    test_data$predicted <- predict(fit)
```

```
    return(test_data)
}

knn_bagging_scenario <- function(train_data, test_data){
    train_model <- function(i){
        train_sample <- sample_data(train_data)
        fit <- kknn(dry_weight ~ ., train = train_sample, test = test_data)
        return(fit)
    }

    predict_function <- function(model){
        return(predict(model))
    }

    return(bag_me(train_data, test_data, train_model, predict_function))
}

knn_scenario_caret <- function(train_data, test_data){
    fit <- train(dry_weight ~ ., data = train_data, method='kknn')
    test_data$predicted <- predict(fit, test_data)

    return(test_data)
}

knn_bagging_scenario_caret <- function(train_data, test_data){
    train_model <- function(i){
        train_sample <- sample_data(train_data)
        fit <- train(dry_weight ~ ., data = train_sample, method='kknn')
        return(fit)
    }

    predict_function <- function(model){
        return(predict(model, test_data))
    }

    return(bag_me(train_data, test_data, train_model, predict_function))
}

svm_scenario_caret <- function(train_data, test_data){

    capture.output(fit <- train(dry_weight ~ ., data = train_data,
                                method='svmLinear3'))

    test_data$predicted <- predict(fit, test_data)

    return(test_data)
}

svm_bagging_scenario_caret <- function(train_data, test_data){
    train_model <- function(i){
        train_sample <- sample_data(train_data)
        capture.output(fit <- train(dry_weight ~ ., data = train_sample,
                                    method='svmLinear3'))
        return(fit)
    }

    predict_function <- function(model){
        return(predict(model, test_data))
    }

    return(bag_me(train_data, test_data, train_model, predict_function, 5))
}

svm_scenario <- function(train_data, test_data){
    tuneResult <- tune(svm, dry_weight ~ ., data = train_data,
                       ranges = list(epsilon = seq(0, 0.2, 0.01), cost = 2^(2:9)))
    fit <- tuneResult$best.model

    test_data$predicted <- predict(fit, test_data)

    return(test_data)
}

svm_bagging_scenario <- function(train_data, test_data){
    train_model <- function(i){
        train_sample <- sample_data(train_data)
        tuneResult <- tune(svm, dry_weight ~ ., data = train_sample,
```

```
                        ranges = list(epsilon = seq(0, 0.2, 0.01), cost = 2^(2:9)))
        fit <- tuneResult$best.model
        return(fit)
    }

    predict_function <- function(model){
        return(predict(model, test_data))
    }

    return(bag_me(train_data, test_data, train_model, predict_function, 5))
}

gbm_scenario_caret <- function(train_data, test_data){
    capture.output(fit <- train(dry_weight ~ ., data = train_data,
                                method='gbm'))
    test_data$predicted <- predict(fit, test_data)
    return(test_data)
}

gbm_bagging_scenario_caret <- function(train_data, test_data){

    train_model <- function(i){
        train_sample <- sample_data(train_data)
        capture.output(fit <- train(dry_weight ~ ., data = train_data,
                                    method='gbm'))
        return(fit)
    }

    predict_function <- function(model){
        return(predict(model, test_data))
    }

    return(bag_me(train_data, test_data, train_model, predict_function))
}

gbm_scenario <- function(train_data, test_data){
    capture.output(fit <- gbm(dry_weight ~ ., data = train_data, n.trees = 1000))
    test data$predicted <- predict(fit, test data, n.trees = 1000)

    return(test_data)
}

gbm_bagging_scenario <- function(train_data, test_data){

    train_model <- function(i){
        train_sample <- sample_data(train_data)
        capture.output(fit <- gbm(dry_weight ~ ., data = train_sample, n.trees =
1000))
        return(fit)
    }

    predict_function <- function(model){
        return(predict(model, test_data, n.trees = 1000))
    }

    return(bag_me(train_data, test_data, train_model, predict_function))
}

prep_output <- function(test){
    cols <- c("dry_weight", "predicted")
    test <- test[,cols]

    return(test)
}

test_model <- function(train_file, test_file, model, reps){
    #Load data
    train_data <- prep_data(train_file)
    test_data <- prep_data(test_file, train_data)

    if (model == "lm"){
        fn <- lm_scenario_caret
    } else if (model == "lm_bag"){
        fn <- lm_bagging_scenario_caret
    } else if (model == "rf"){
        fn <- random_forest_scenario_caret
    } else if (model == "ann"){
```

```
        fn <- ann_scenario
    } else if (model == "ann_bag"){
        fn <- ann_bagging_scenario
    } else if (model == "glm"){
        fn <- glm_scenario
    } else if (model == "knn"){
        fn <- knn_scenario_caret
    } else if (model == "knn_bag"){
        fn <- knn bagging scenario caret
    } else if (model == "svm"){
        fn <- svm_scenario_caret
    } else if (model == "svm_bag"){
        fn <- svm_bagging_scenario_caret
    } else if (model == "gbm"){
        fn <- gbm scenario caret
    } else if (model == "gbm_bag"){
        fn <- gbm_bagging_scenario_caret
    }

    test <- suppressWarnings(fn(train_data, test_data))
    test <- prep output(test)
    return(test)
}
```

8.2 Scatter search method, first experiment additional data

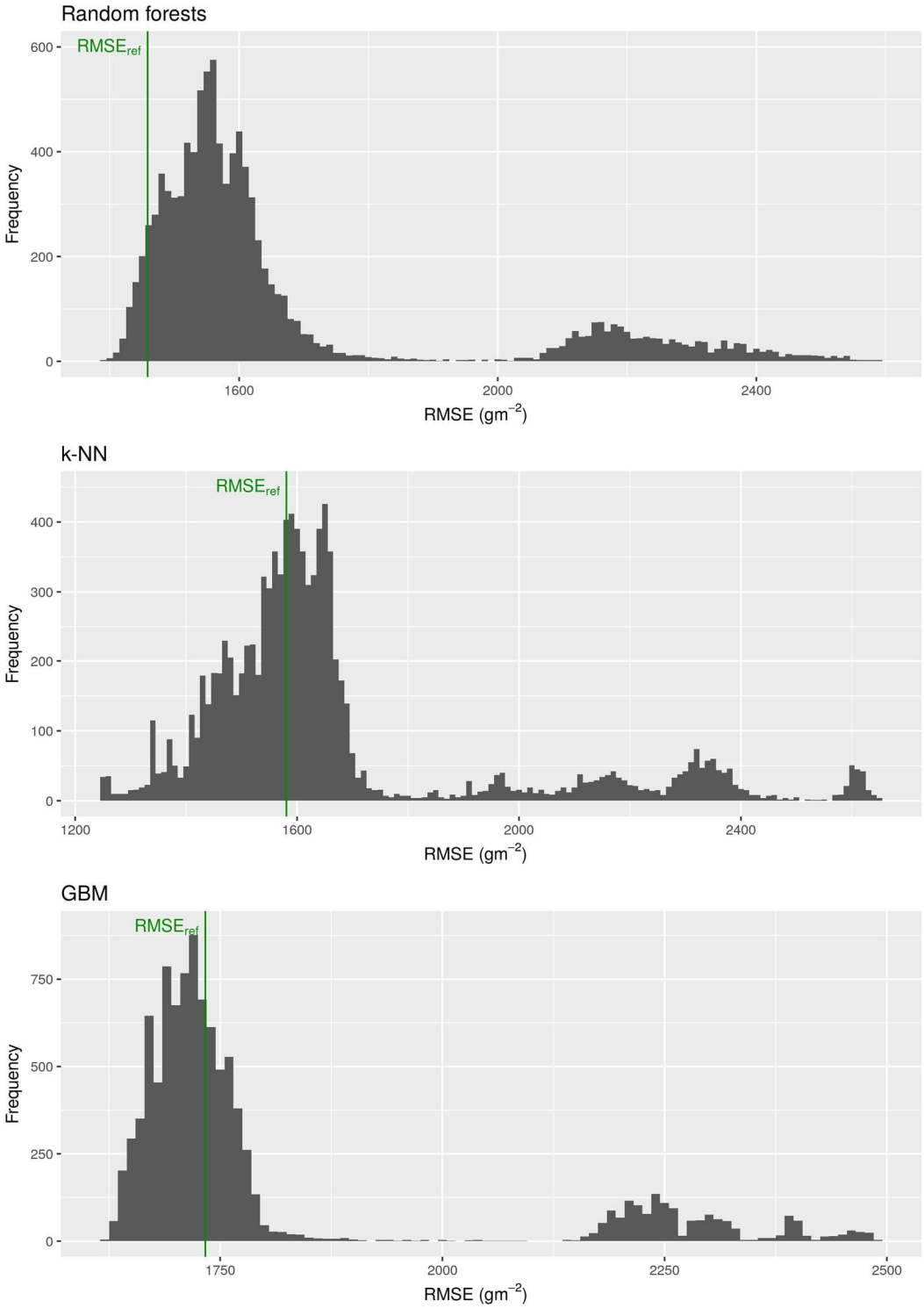

Figure 8.1 Histogram of the train RMSE of the models from 10000 data collection strategies, sampled randomly from the strategies database from the first scatter search experiment in chapter 5 (section 5.3.1). The reference model RMSE is shown as a green line. The RMSE of the models was calculated using cross-validation on the training ABR61 dataset, years 2011-2015.

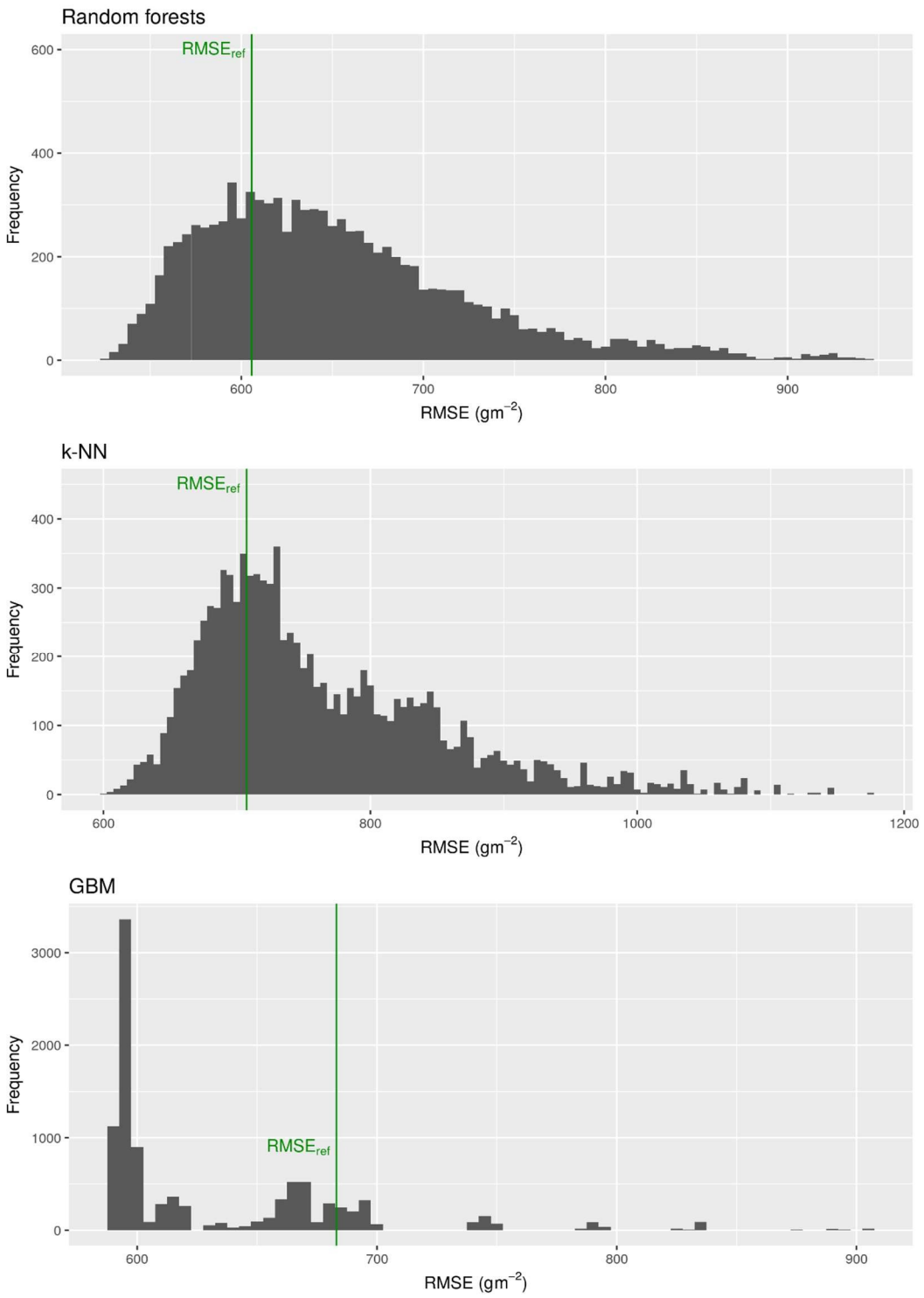

Figure 8.2 Histogram of the test RMSE of the models from 10000 data collection strategies, sampled randomly from the strategies database from the first scatter search experiment in chapter 5 (section 5.3.1). The reference model RMSE is shown as a green line. The models were tested against the unseen ABR61 test dataset (year 2016). The RMSE of the models was calculated using predicted and actual dry weight values.

8.3 Scatter search method, second experiment, additional data

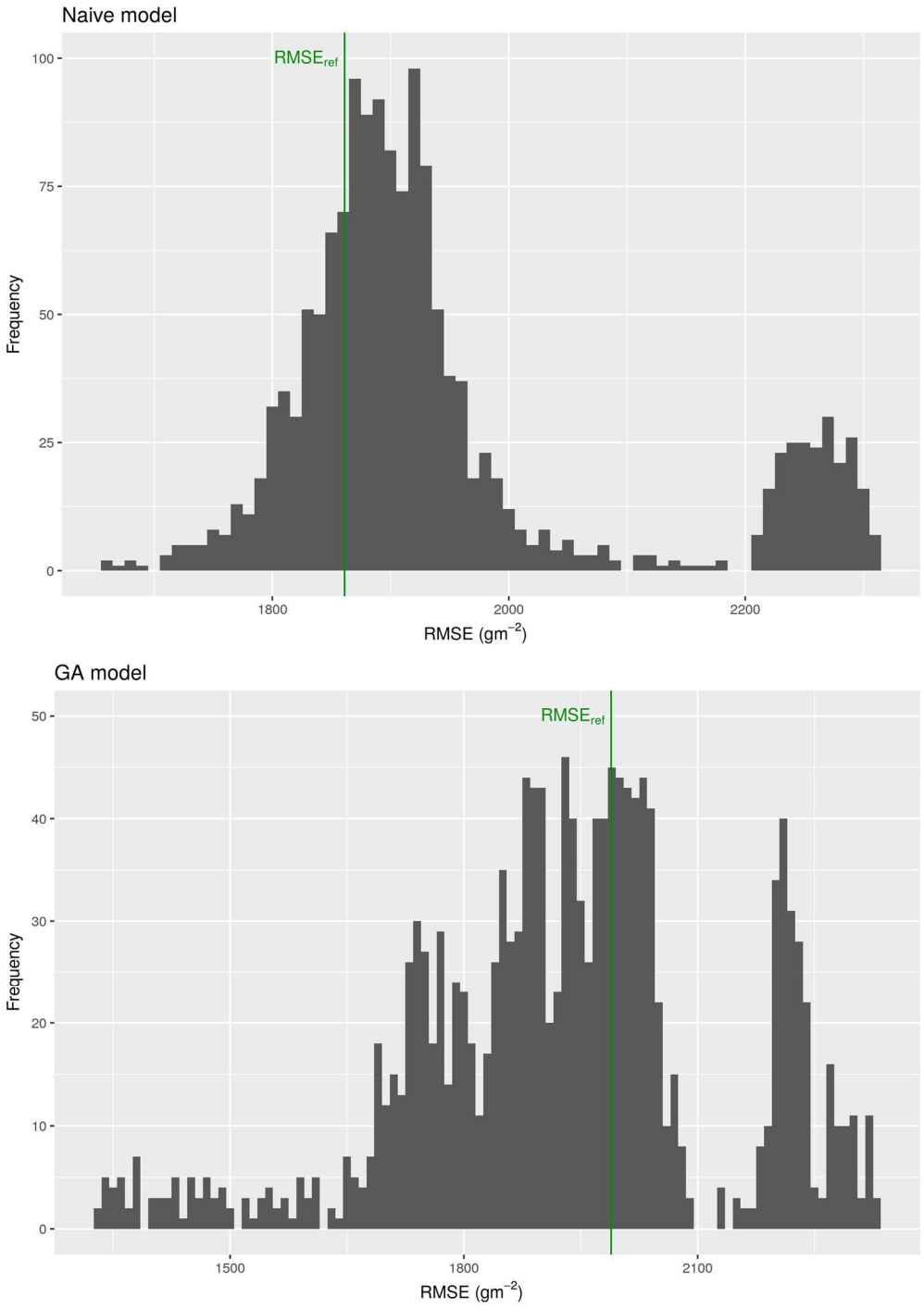

Figure 8.3 Histogram of the train RMSE of the models from 1500 data collection strategies, sampled randomly from the strategies database from the second scatter search experiment in chapter 5 (section 5.3.2). The reference model RMSE is shown as a green line. The RMSE of the models was calculated using cross-validation on the training ABR61 dataset, years 2011-2015.

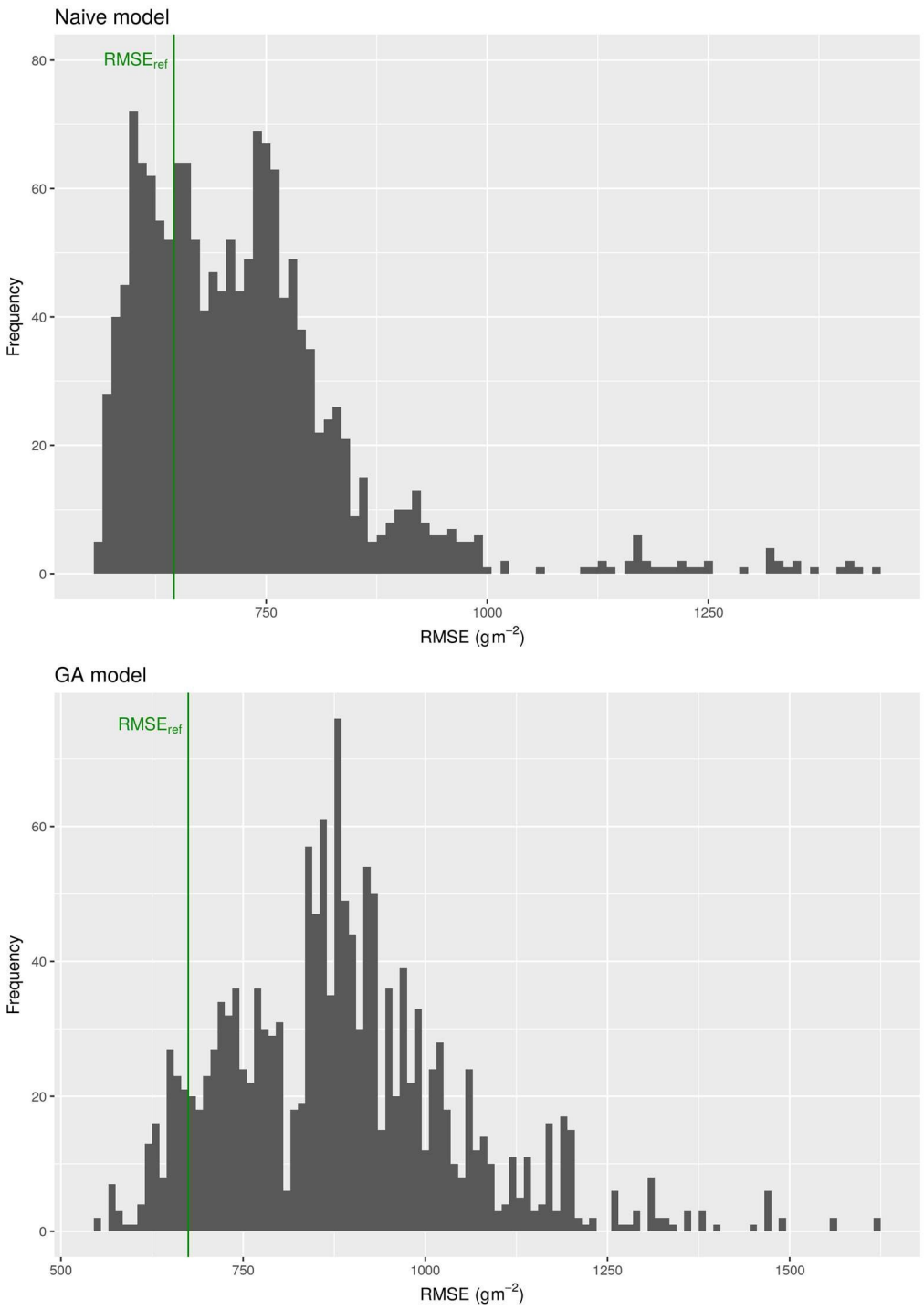

Figure 8.4 Histogram of the test RMSE of the models from 1500 data collection strategies, sampled randomly from the strategies database from the second scatter search experiment in chapter 5 (section 5.3.2). The reference model RMSE is shown as a green line. The models were tested against the unseen ABR61 test dataset (year 2016). The RMSE of the models was calculated using predicted and actual dry weight values.